Filipe Miguel Guerreiro Martins

Licenciado em Ciências da
Engenharia Electrotécnica e de Computadores

eVentos 2 - Autonomous sailboat control

Dissertação para obtenção do Grau de Mestre em
Engenharia Electrotécnica e de Computadores

Orientador : Prof. Doutor Luís Filipe dos Santos Gomes,
Professor Associado, Universidade Nova de Lisboa

Júri:

Presidente:	Mário Fernando da Silva Ventim Neves
Arguente:	Luís Filipe Figueira Brito Palma
Vogal:	Luís Filipe dos Santos Gomes

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2013

eVentos 2 - Autonomous sailboat control

Acknowledgements

I would like to start by thanking my dissertation supervisor, Prof. Luis Gomes for all his help, patience and support when i needed and for putting up with my work methods. I would also like to express my gratitude to my laboratory colleagues Marcelo Rodrigues and Flávio Gil for all their support and help.

My next acknowledgements go to all my academic colleagues and friends that made my academic path much more interesting and worthwhile, namely Pedro Cunha, Pedro Sardinha, Carlos Ribeiro, Diogo Morgado, João Silva, Nuno Pereira, Hugo Serra, Pedro Leitão, José Vieira, António Furtado, Ricardo Mendonça, Pedro Mota and many, many more.

I'd like to also acknowledge all my friends and life-time partners from the university's archery team, for being fun (most of the times even moronic) and for all their support when i really needed it.

Last but not least my gratitude goes to my parents and family for their continuous support along these academic years.

Abstract

Sailboat navigation started as a way to explore the world. Even though performance is significantly lower than that of a motorboat, in terms of resources, these vessels still are the best low-cost solutions. On the past, navigation depended greatly on estimates or on the stars. Nowadays it depends on precise data provided by a variety of electronic devices, independent from the user's location.

Autonomous sailboats are vessels that use only the wind for propulsion and have the capacity to control its sails and rudders without human intervention. These particularities give them almost unlimited autonomy and a very valuable ability to fulfill long term missions on the sea, such as collecting oceanographic data, search and rescue or surveillance.

This dissertation presents a fuzzy logic controller for autonomous sailboats based on a proposed set of sensors, namely a GPS receiver, a weather meter and an electronic compass. Following a basic navigation approach, the proposed set of sensors was studied in order to obtain an effective group of variables for the controller's fuzzy sets, and rules for its rule base. In the end, four fuzzy logic controllers were designed, one for the sail (to maximize speed) and three for the rudder (in order to comply with all navigation situations). The result is a sailboat control system capable of operation in a low cost platform such as an Arduino prototyping board. Simulated results obtained from a data set of approximately 100 tests to each controller back up the theory presented for the controller's operation, since physical experimentation was not possible.

Keywords: Autonomous sailboat, fuzzy logic control, fuzzy set, rule base, sailing, Arduino, GPS, Compass, Anemometer

Resumo

A navegação à vela estreou-se como uma maneira de explorar o mundo. Mesmo tendo um rendimento bastante menor que o de um barco a motor, estes navios continuam a ser a melhor solução de baixo custo no que toca ao gasto de recursos. No passado, a navegação dependia grandemente de estimativas e da leitura das estrelas, hoje em dia depende de dados precisos fornecidos por uma variedade de aparelhos electrónicos, independentemente da localização do utilizador.

Veleiros autónomos são veículos que apenas utilizam o vento como forma de propulsão, com capacidade de controlar as suas velas e lemes sem ajuda humana. Estas particularidades dão autonomia quase ilimitada e uma valiosa habilidade de cumprirem missões longas, tais como recolha de dados oceanográficos, busca e salvamento ou vigilância.

Esta dissertação apresenta um controlador difuso para veleiros autónomos baseado num conjunto proposto de sensores, nomeadamente um receptor GPS, anemómetro, catavento e um sensor de bússola. Seguindo uma abordagem baseada em navegação simples, o conjunto de sensores proposto foi estudado de forma a obter um grupo eficaz de variáveis para os conjuntos difusos do controlador, e de regras para a sua base de regras. No final, quatro controladores de lógica difusa foram projectados, um para a vela (de forma a maximizar a velocidade) e três para o leme (de forma a cumprir todas as situações possíveis de navegação). O resultado foi a criação de um sistema de controlo para navegação à vela capaz de actuar numa plataforma de baixo custo, uma placa de prototipagem Arduino. Os resultados de simulação obtidos de um conjunto de dados de aproximadamente 100 testes a cada controlador apoiam a teoria apresentada para a operação deste controlador, visto que a experimentação em ambiente físico não foi possível.

Palavras-chave: Veleiro autónomo, controlador de lógica difusa, conjunto difuso, base de regras, navegação à vela, Arduino, GPS, Bússola, Anemómetro

Contents

Acknowledgements **v**

Abstract **vii**

Resumo **ix**

List of Figures **xv**

List of Tables **xix**

Abbreviations **xxi**

1 Introduction **1**
1.1 Motivation 1
1.2 Objectives and contributions 2
1.3 Dissertation Outline 2

2 Related work **5**
2.1 FASt - FEUP Autonomous Sailboat 6
2.1.1 Hardware 6
2.1.2 Software 8
2.2 Fuzzy logic control on autonomous sailing systems 9
2.2.1 System Architecture 9
2.2.2 Fuzzy Control System 10
2.2.3 Results and conclusions 12

3 Supporting Concepts **15**
3.1 Basics of Sailing 15
3.1.1 The vessel 15
3.1.2 The use of sails 17
3.1.3 Sailing Techniques 18

3.2 Fuzzy Control . . . 21
3.2.1 An Introduction . . . 21
3.2.2 Applications . . . 22
3.2.3 Essential characteristics . . . 23
3.2.4 Fuzzy logic controller architecture . . . 24

4 Software and Hardware **29**
4.1 Arduino . . . 29
4.1.1 Hardware . . . 30
4.2 Proposed sensors for the project . . . 31
4.2.1 Weather meter . . . 31
4.2.2 CMPS10 Tilt Compensated Magnetic Compass . . . 32
4.2.3 EM-406A SiRF III GPS Receiver with Antenna . . . 32
4.3 Available sensors and emulators . . . 34
4.3.1 Simulation platform . . . 34
4.3.2 PIC-Arduino interface . . . 35
4.4 Xfuzzy . . . 35

5 Proposed controller **37**
5.1 Transition between manual and automatic control . . . 38
5.2 Designed controller . . . 41
5.3 Controller Variables . . . 42
5.3.1 Wind Alignment . . . 42
5.3.2 Directional Alignment . . . 44
5.4 Control structure . . . 46
5.4.1 Sail . . . 47
5.4.2 Rudder . . . 50
5.4.3 Rudder decider structure . . . 55
5.4.4 Goal approaching . . . 58

6 Experimental results **63**
6.1 Xfuzzy- Arduino integration . . . 64
6.2 Experimental results and analysis . . . 64
6.2.1 Sail . . . 65
6.2.2 Favorable Wind . . . 68
6.2.3 Tack Left . . . 71
6.2.4 Tack Right . . . 73

7 Conclusions and future work **77**
7.1 Conclusions . . . 77
7.2 Future Work . . . 78

Bibliography **81**

A Experimental results tables **87**

List of Figures

2.1 FEUP Autonomous Sailboat [AC09]. 6
2.2 FASt's electronic system, adapted from [AC09]. 7
2.3 FASt's software organization, adapted from [AC09]. 8
2.4 (a) - Robbe Atlantis yacht; (b) - Detailed system architecture; 10
2.5 Fuzzy sets for rudder input. 10
2.6 Fuzzy set for rudder output. 11
2.7 Fuzzy set for sail input. 11
2.8 Fuzzy set for sail output. 12
2.9 Test runs for turning against and in favor of the wind respectively with 2 second time intervals. 13

3.1 Main sailboat equipment [Boa] and side terms, viewed from above. 16
3.2 Killer Whales Yacht. 17
3.3 Experienced winds [Ass]. 17
3.4 Lift and aerodynamic force created by the wind [Saib]. 19
3.5 Sailing techniques 1 [Sal96]. 19
3.6 Sailing techniques 2 [Sal96]. 20
3.7 Sailing techniques 3 [Cat, Sal96]. 21
3.8 Fuzzy logic controller [Alo]. 24
3.9 Different membership function types: (a) - Triangular ; (b) - Trapezoidal; (c) - Bell-shaped; (d) - Singleton; 25
3.10 Fuzzification stage for crisp inputs [Yal]. 25
3.11 Mamdani two input, two rule inference system [Kna]. 27

4.1 Arduino Mega 2560 board [Ban]. 30
4.2 Proposed sensors: a)Weather Meter [Sysb]; b)Compass Module [PTRb]; c)GPS Module [PTRa]; 33
4.3 Sensorial emulation setup [Gil13]. 34
4.4 Simulation interface, adapted from [Gil13]. 35

5.1 Signal generated when the remote controller is off. 37
5.2 Signal generated when the remote controller is off. 38
5.3 Signal generated when the remote controller is on (noise is irrelevant in this case as it is only generated from the RF waves into the oscilloscope). . 38
5.4 Transition between manual and remote control. 39
5.5 Different characteristics in the controller's signal. 39
5.6 a)Counter implementation flowchart; b)Transition strategy flowchart; . . . 40
5.7 Pin scheme used for transition between controls. 41
5.8 General layout of inputs and outputs of the controller system. 42
5.9 New set of variables used for sail and rudder control. 42
5.10 Wind alignment value range. 43
5.11 Wind alignment algorithm confirmation. 43
5.12 Original graphical projection of the intended course. 44
5.13 Modified graphical projection of the intended course. 44
5.14 Rectangular to polar coordinate conversion, adapted from [Vau]. 45
5.15 Directional alignment value range. 45
5.16 Updated layout of the controller system. 47
5.17 Trapezoid point specification. 47
5.18 Roll's fuzzy set. 48
5.19 Wind Speed's fuzzy set. 48
5.20 Wind Alignment's fuzzy set. 49
5.21 Sail's fuzzy set. 49
5.22 Directional Alignment's fuzzy set. 51
5.23 Boat Speed's fuzzy set. 51
5.24 Rudder's fuzzy set. 52
5.25 State machine of the rudder decider structure. 56
5.26 Potentially used sailing area, defined in the beginning of the course. . . . 57
5.27 Radius from both the objective buoy and distance to the vessel. 58
5.28 a) Finish line half planes identification; b) Lines at right angles to the finish line; 59
5.29 Zone division for finish line detection. 60

6.1 Experimental results for the sail controller part 1. 65
6.2 Experimental results for the sail controller part 2. 65
6.3 Experimental results for the sail controller part 3. 66
6.4 Experimental results for the sail controller part 4. 66
6.5 Experimental results for the sail controller part 5. 67
6.6 Experimental results for the favorable wind controller part 1. 68
6.7 Experimental results for the favorable wind controller part 2. 68
6.8 Experimental results for the favorable wind controller part 3. 69
6.9 Experimental results for the favorable controller part 4. 70

6.10 Experimental results for the favorable wind controller part 5. 70
6.11 Experimental results for the tack left controller part 1. 71
6.12 Experimental results for the tack left controller part 2. 71
6.13 Experimental results for the tack left controller part 3. 72
6.14 Experimental results for the tack left controller part 4. 72
6.15 Experimental results for the tack right controller part 1. 73
6.16 Experimental results for the tack right controller part 2. 73
6.17 Experimental results for the tack right controller part 3. 74
6.18 Experimental results for the tack right controller part 4. 74

7.1 Alternative tacking approach [Saia]. 78

List of Tables

2.1 Rudder rule base. 11

3.1 Applications of fuzzy logic control, as seen in [YLZ95] page 6. 22

4.1 Arduino Mega 2560 features. 31
4.2 RMC Data Format, adapted from [ST]. 33

5.1 Test results for acquiring efficient threshold values. 40
5.2 Confirmation examples for the directional alignment algorithm. 46
5.3 Trapezoid values for each fuzzy set on the sail controller. 50
5.4 Fuzzy rules for the sail inference system. 50
5.5 Trapezoid values for each fuzzy set on the rudder controllers. 52
5.6 Fuzzy rules for the rudder inference system - favorable wind situation. . . 53
5.7 Fuzzy rules for the rudder inference system - left tack situation. 54
5.8 Fuzzy rules for the rudder inference system - right tack situation. 55

7.1 Available router model research. 79

A.1 Sail controller experimental results. 87
A.2 Rudder: Favorable Wind controller experimental results. 90
A.3 Rudder: Tack Right controller experimental results. 92
A.4 Rudder: Tack Left controller experimental results. 94

Abbreviations

COM	**C**omponent **O**bject **M**odel
CPU	**C**entral **P**rocessing **U**nit
FEUP	**F**aculdade de **E**ngenharia da **U**niversidade do **P**orto
FPGA	**F**ield **P**rogrammable **G**ate **A**rray
GPS	**G**lobal **P**ositioning **S**ystem
I2C	**I**nter **I**ntegrated **C**ircuit
ICSP	**I**n **C**ircuit **S**erial **P**rogramming
JRE	**J**ava **R**untime **E**nvironment
NMEA	**N**ational **M**aritime **E**lectronics **A**ssociation
PCB	**P**rinted **C**ircuit **B**oard
PWM	**P**ulse **W**idth **M**odulation
RAM	**R**andom **A**ccess **M**emory
RF	**R**adio **F**requency
RMB	**R**ecommended **M**inimum Navigation Information
RMC	**R**ecommended **M**inimum **D**ata
SPI	**S**erial **P**eripheral **I**nterface
SSI	**S**erial **S**ynchronous **I**nterface
UART	**U**niversal **A**synchronous **R**eceiver/**T**ransmmiter
USB	**U**niversal **S**erial **B**us

1

Introduction

1.1 Motivation

Developing an autonomous sailboat is a relatively complex task, from the vessel's mechanical construction to implementing its electronic systems. Sailing can be viewed as a complex problem dependent on unpredictable environmental conditions and interrelated variables such as the designated course, wind (direction and speed) and sea state. In conventional sailing, the sailor controls the rudder according to the desired course of action and the sail to maximize velocity. For a given course, there is always an optimum angle between the sail and direction of the wind to maximize the speed of the vessel.

Autonomous sailboats are robotic vessels that use wind for propulsion and are capable of controlling the rudder and sails without human intervention. Currently, international regulations for navigation do not contemplate unmanned vehicles because, in practical terms, solutions that carry out the International Maritime Organization (IMO) regulations are not yet available. With the potential for autonomous vessels to carry out real missions, such as navigation for single handed sailors [War91], ocean sampling [NAC08] or simple auto-pilot functions, allied with low cost solutions can help change this scenario. Various authors addressed the issues associated with autonomous sailing. The basic course control problem (resorting on a rudder) has been partially solved for decades and is currently used on commercial and pleasure crafts. However, the performance of a sailboat is strongly dependant on the way the rudder is controlled in order to advance using minimal wind variations. On competitions, resorting to autopilot is deemed as a "necessary evil". This way, it is obvious that room exists for new developments and various projects exist in the area, where techniques such as neural networks [ATX01] and fuzzy control [SPJ07] have been applied.

1.2 Objectives and contributions

Completely autonomous navigation requires other tasks such as reaching a pre-determined set of geographic points, maintaining the vessel inside a delimited area or achieving minimal time to pass through a designated course. Various works have been published [ATA02, SP08, EK06] and from these efforts resulted autonomous sailboats demonstrating these technologies, appearing in competitions such as Microtransat [Micc]. In November 2010, a meeting between various Portuguese academic institutions culminated in the idea to start a new maritime robotic competition, MAROCUP (MAritime RObotic CUP). This dissertation's main objective is to design, simulate and implement the general architecture for UNL's sailboat controller. To achieve this objective, comprehension of how a sailboat works is necessary to obtain an expert's view of the system. The strategy used was independent control of rudder and sail, for decreased complexity and the type of control used was fuzzy logic. This type of control was chosen mainly because of its ability to design and simulate a system without a proper description. In this case, the sailboat system had no description to start working with. Rudder control needed more than one controller since steering a sailboat is wind dependent, making it a delicate and complex system, as seen ahead. A secondary objective of this dissertation was improving the work already done in this project [Gil13]. This objective was achieved by designing a method for transition between the vessel's automatic and manual control modes and by various suggestions done in chapter 7.

1.3 Dissertation Outline

Besides this introductory chapter, this dissertation is organized in six more chapters.

Chapter 2 gives a small introduction about various other projects developed in the area. A mora detailed description is given of two points of interest: an autonomous sailboat system and a fuzzy logic controller for sailing systems. The sailboat was designed in FEUP and its hardware and software specifications are presented. The fuzzy control system is described and it is important to notice that it was experimented on a vessel much similar to the one designed by FEUP.

Chapter 3 presents the main concepts related with the work done on this dissertation. First, a basic introduction to a sailboat's most important components, use of sails and techniques used is presented. A short introduction about fuzzy control is also given, presenting some applications, its essential characteristics and how a typical inference system works.

Chapter 4 describes the hardware and software for the project. In hardware aspects, the Arduino prototyping board is described. Also, proposed and available sensors for development are presented, namely a weather meter, magnetic compass, GPS receiver and emulators. These are not the same since most of the work done was in simulation because of lack of materials. In terms of software, Sevilla Microelectronic Institution's

Xfuzzy design tool is described.

Chapter 5 describes all the developed work. After a general layout, a description of the transition method between automatic and manual control is given before presenting the designed controller. This controller is thoroughly described, its variables, control strategies, methods for changing strategies and two techniques for approaching a finish point.

Chapter 6 presents experimental results obtained from both Xfuzzy's simulation and Arduino code for the designed controllers. These experimental results are obtained using MATLAB to graphically obtain representations of the the data sets.

Chapter 7 collects a set of conclusions and further research opportunities for this project.

2

Related work

A sailboat is a strongly non-linear system that has been given proof to be easily controlled. Its mechanical design comes from hundreds of years of evolution concerning mainly the efficiency and reliability of the vehicle and its ease of use by human operators. Out of many, this chapter introduces an existing autonomous sailboat system and also a fuzzy logic controller designed for use in sailboats.

Additional systems are described in [BJ12, KSJM09, SJ12, RRGI+11, Bri11]. [BJ12] describes a simple controller based on an expert's approach and tested on the autonomous sailboat VAIMOS built by IFREMER. [KSJM09] describes a modification to another autonomous system (ROBOAT) to develop passive acoustic monitoring of marine mammals. [SJ12] describes the above system ROBOAT. [RRGI+11] describes a sailing fuzzy logic system based on an omni-directional camera for obstacle detection. [Bri11] describes yet another autonomous sailboat system, from the prototype to the control solution.

2.1 FASt - FEUP Autonomous Sailboat

The autonomous sailboat FASt shown in figure 2.1 is an unmanned vessel $2,5$ meters wide, designed and developed in FEUP (Faculdade de Engenharia da Universidade do Porto, Portugal) as an extra-curricular project by students of master's degree in electrical engineering. Its rig is similar to that of real manned sailboat and it entered the first edition of the World Robotic Sailing Championship in 2008[OGfiC].

Figure 2.1: FEUP Autonomous Sailboat [AC09].

Its electronic control and navigation system is based on a reconfigurable platform containing a FPGA (Field Programable Gate Array) executing the operating system uCLinux in a Microblaze processor. Besides the CPU and the peripherals necessary to the computational system, the FPGA implements a set of dedicated interface and processing units that realize various communication processes with the peripherals (sensors and actuators). These modules give relevant and pre-processed data do the CPU from each sensor and generate the control signals necessary to act on the electrical motor actuators) [AC09].

2.1.1 Hardware

FASt's electronic system was built around a FPGA based commercial platform, which implements a computational system. Besides the incorporated sensors and actuators, the vessel has an Ethernet router with wireless connection, remote radio controller and an IRIDIUM modem for short data message communication when sailing in open sea. The electric power necessary onboard is provided be a 45 Peak-Watt solar panel, two lithium-ion with a total capacity of 190 Watt/hour and a commercial module which integrates the battery charger and power supply.

Figure 2.2 shows a simplified diagram of the electronic system included in the vessel.

A motherboard aggregates all the electronic components used in interfacing with the FPGA. It also offers a set of connectors where all of FASt's sensors and actuators are linked. Also, it is included a SD card reader which is used for data registry and non-volatile data such as real time clock, used to maintain the main state variables and to provide the current time whenever the GPS module is put in low-consumption mode.

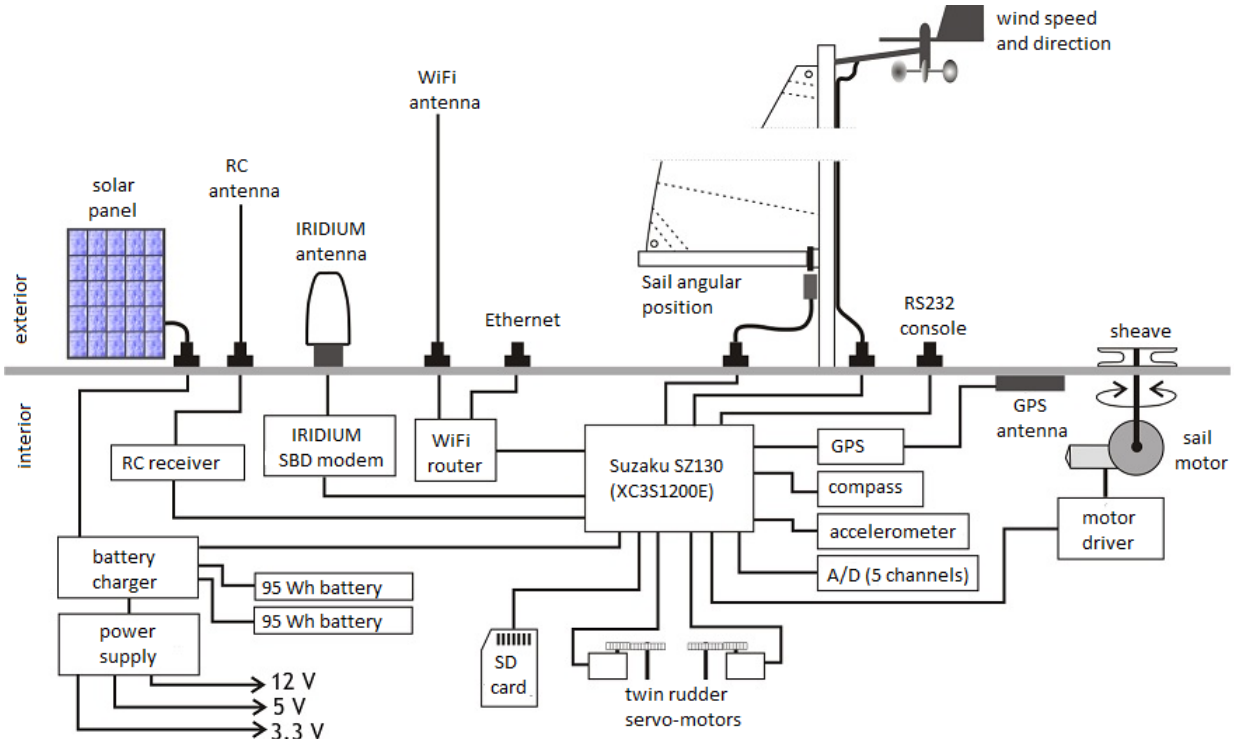


Figure 2.2: FASt's electronic system, adapted from [AC09].

The computational system was implemented around a Spartan3E 1200 (Suzaku SZ130 [AT]) FPGA. The platform has 32 Mega-Bytes for SDRAM, 8 Mega-Bytes for flash memory, Ethernet connection and a serial port for the data console. The flash memory maintains an image of the operating system and the FPGA configuration. This information can be refreshed from the operating system's shell. There is a total of 86 terminals directly connected to FPGA pins, which permits the direct connection of various peripherals.

The sensors and actuators present in the system connect to these pins through isolation circuits. The main sensors are a GPS receptor, electronic compass, inclinometer, anemometer, wind vane and sail angle detector. There are still 5 channels available for digital to analog conversion in case of expansion. The actuators allow the control of the twin rudders and the position of the two available sails. Rudder control is done by 2 independent servo-motors. The position of the sails is controlled by a single motor by a PWM (Pulse Width Modulation) modulator.

2.1.2 Software

The software component was developed in C, using standard Linux libraries also available in uCLinux. The navigation and control processes are divided in various concurrent processes communicating between themselves through UDP sockets, as shown in figure 2.3. The adoption of an operating system based in Linux allows an highly efficient software development environment. Any of the implemented programs can be compiled and executed in any Linux machine that is accessible via TCP/IP to FASt's onboard computer.

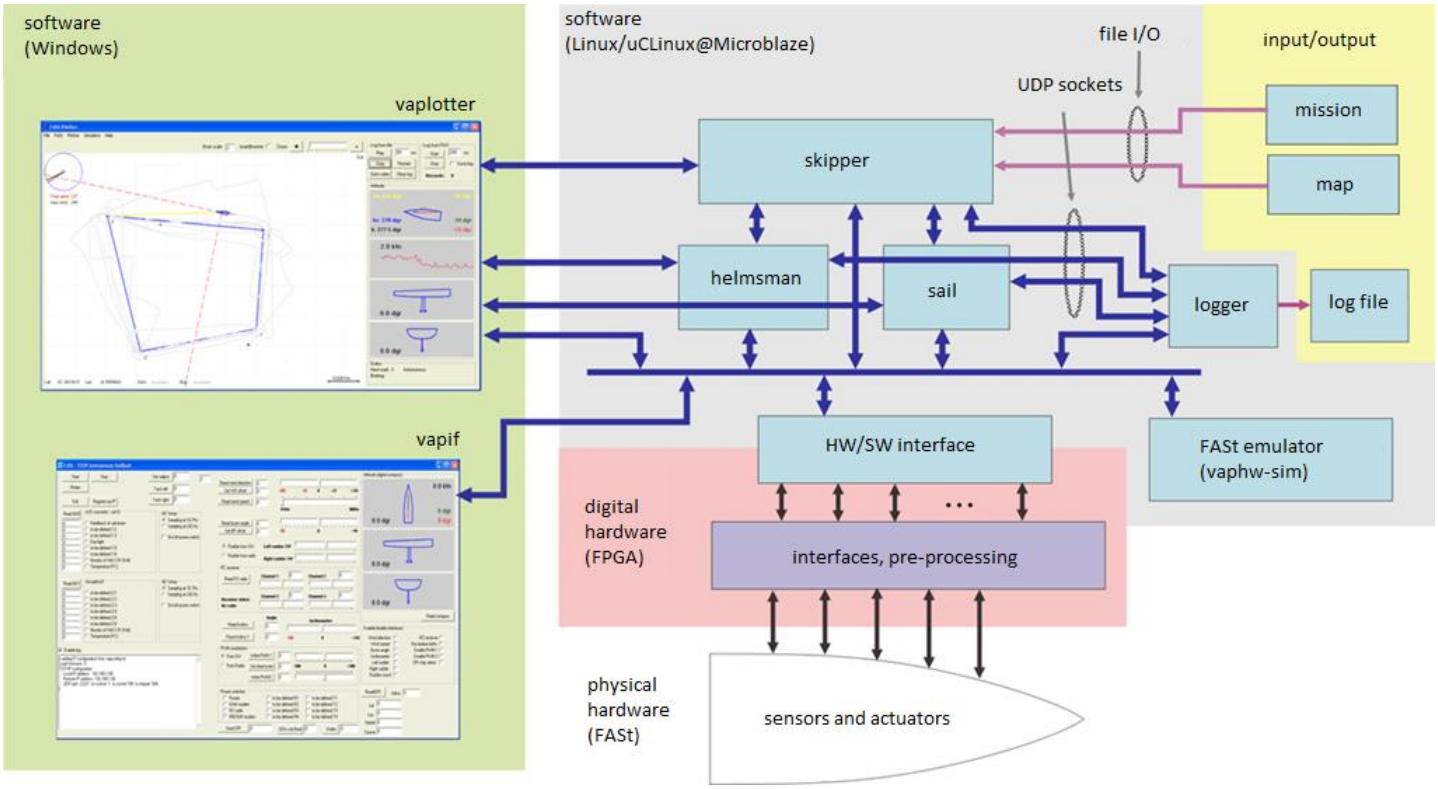


Figure 2.3: FASt's software organization, adapted from [AC09].

To support the validation and development of the software, a FASt emulator was developed in laboratory replacing the layer that implements the hardware interface. The emulator (`vaphw-sim`) includes a non-linear dynamic model of the vessel, which simulates its behavior according to the wind and the sail and rudder positioning commands. This is an essential step to allow the independent development of the software part, free from the hardware and to limit the number of necessary field tests, which require significant logistics, resource and time spending.

In order to ease the troubleshooting of the interface software and developed hardware an interactive visual application was created. This application allows direct access to all FASt's peripherals (`vapif`). Real time monitorization is made by another application (`vaplotter`) which graphically registers the vessels position, including all the relevant parameters for the navigation and control algorithms. The same application allows the vessel's mission programming as a sequence of geographic points that should be visited,

and also the reproduction of saved data during a navigation mission, real or simulated.

2.2 Fuzzy logic control on autonomous sailing systems

Many different systems are available that can assist in the steering of a sailboat. The most popular commercial systems are autopilots and wind vanes and both systems can keep the vessel on a predefined course. While the autopilot keeps the vessel on course using compass data, the wind vanes keep the course using the wind. Both systems control the rudder but have no influence on the sail sheets, these need to be adjusted manually. Many methods for rudder control exist, fuzzy [ASC97, Van97] as well as nonfuzzy [War91] but none of them cover sail control.

The system proposed in this section is able to control all the manoeuvres of an autonomous sailboat. A separate software module is responsible for weather routing and delivering directions for the actual vessel position and weather conditions in real time. If the vessel's direction deviates, the system adjusts the rudder in order to achieve the desired course. A second control system assures that there is flow in the sails in order to get speed. It also controls the tilting of the vessel, depending on the speed and direction of the wind. The main aim of this system is to imitate the behavior of an experienced human sailor. Therefore it is not limited to a specific vessel, but applicable to every common type of sailboat. Fuzzy logic is used to control both actuators for sail and rudder. This is a very suitable method for transforming expert's knowledge into a computer program in form of if-then rules, as presented in [SPJ07], which will be described in the following sub-sections.

2.2.1 System Architecture

Experimenting the system was carried out on the 1,38m yacht model "Robbe Atlantis" (figure 2.4 (a)) under real-world conditions. The vessel won the First Microtransat Challenge [Micc] for autonomous sailboats in Toulouse, France in June 2006. The aim was to demonstrate completely autonomous sailing, where routing and navigation have to run automatically on the vessel. The "Robbe Atlantis" is usually used as a remote controlled sailboat. For testing purposes it was additionally equipped with various sensors to measure the environmental conditions. A program called "abstractor" running on the boat gathers sensor data and transforms it into semantically useful information for the routing software. The modular architecture of the system is shown in figure 2.4 (b).

Various sensors deliver data for the routing (navigator) and fuzzy inference system (skipper) which covers the rudder and sail control circuits. The data abstraction layer transforms raw sensor data into useable data for the high level applications. Both control circuits periodically send their results (rudder and sail position) back to the abstraction layer. This data is then transformed back into a format readable by the hardware controller devices of the actuators.

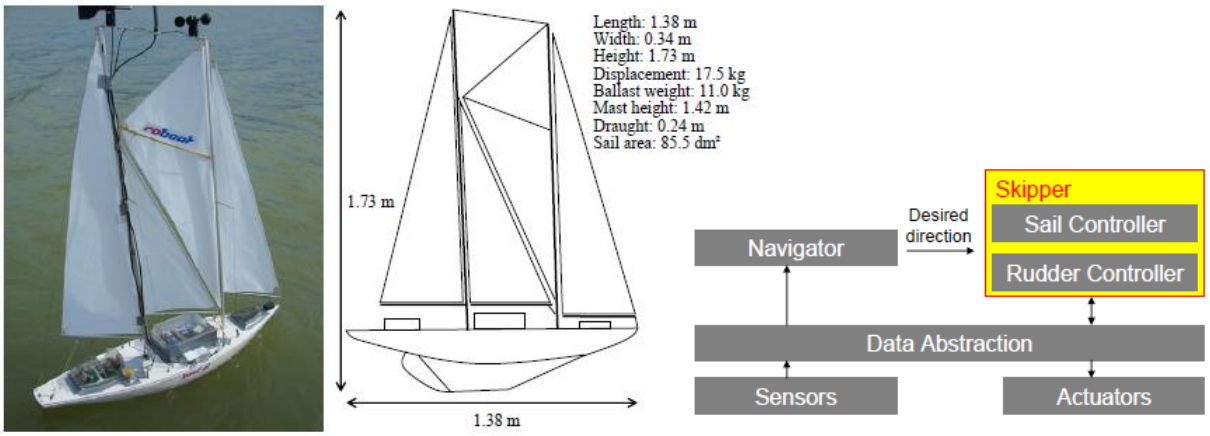


Figure 2.4: (a) - Robbe Atlantis yacht; (b) - Detailed system architecture;

2.2.2 Fuzzy Control System

In sailing, different persons are able to control the rudder and sail separately, without communication. Therefore in the presented system, two independent working control loops are responsible for the rudder and sail actuators. The rudder controller keeps the vessel on a predefined course given by the routing software. The sail controller avoids capsize and assures that there is enough flow in the sails, giving propulsion power to the vessel. Both actuators should be controlled fast but smoothly, without leaps or oversteering. Two Sugeno type fuzzy inference systems were used to reach this goal. Trapezoid fuzzy sets are used as inputs and singletons represent output variables. Defuzzification is done by the center of gravity of singletons method. The execution of the system was identified by experimentation and has to be adapted for every type of vessel.

The current vessel direction and desired direction are the input data for the rudder control circuit. The difference between these two give the necessary course correction which enters directly into the fuzzy system (desired direction). To avoid oversteering, the angular velocity of the vessel flows as an additional input variable (turn). The fuzzy sets representing the linguistic terms of these input variables are shown in figure 2.5.

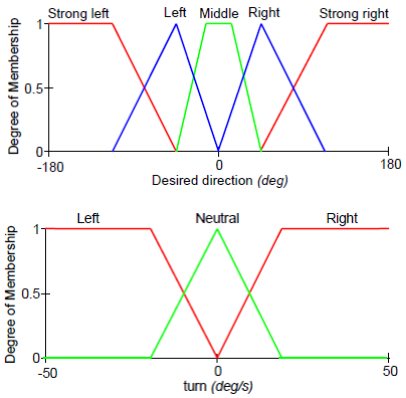


Figure 2.5: Fuzzy sets for rudder input.

The rudder control output is the change of the rudder position. The fuzzy variable rudder change (in percentage) contains five singletons representing the linguistic terms of the variable (figure 2.6).

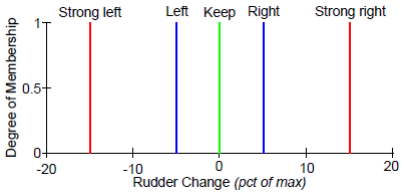


Figure 2.6: Fuzzy set for rudder output.

The rule base in table 2.1 of the rudder control systems contains 15 rules in the form IF *desired direction* IS x AND *turn* IS y THEN *rudder change* IS z .

Table 2.1: Rudder rule base.

Rudder change	Turn		
Desired direction	Left	Neutral	Right
Strong left	Left	Strong left	Strong left
Left	Keep	Left	Strong left
Middle	Right	Keep	Left
Right	Strong right	Right	Keep
Strong right	Strong right	Strong right	Right

The tilting of the vessel, direction and speed of the wind are the inputs for the sail control circuit. The sail fuzzy system calculates direction and amount of adjustment necessary for the sail winch. The aim of this control system is to keep the vessels tilting on an optimum according to the actual wind conditions.

The variable heeling(deg) acts as an input for the sail inference system, which is the difference between the desired tilting and the actual tilting of the vessel. The fuzzy set representing the linguistic terms of this input variable is shown in figure 2.7.

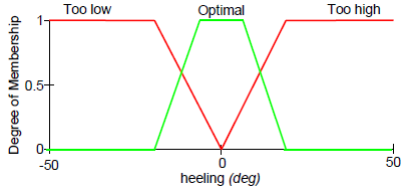


Figure 2.7: Fuzzy set for sail input.

The sail control system output is the change of the sheet position, via the sheet winch. The fuzzy variable sail change (in percentage) contains three singletons representing the linguistic terms of the variable (figure 2.8)

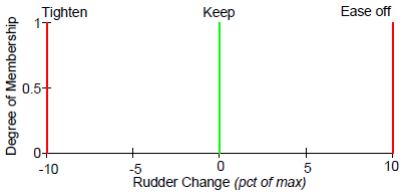


Figure 2.8: Fuzzy set for sail output.

The rule base of the sail control system contains the following if-then rules:

- If heeling is *too low* then *tighten* sheets;
- If heeling is *optimal* then *keep* sheets;
- If heeling is *too high* then *ease off* the sheets;

2.2.3 Results and conclusions

Several test runs were carried out to demonstrate the feasibility and suitability of the presented approach. The data presented on figure 2.9 refers to the final test run prior to the Microtransat competition, where wind conditions were within operation range of the demonstration vessel. The process of two maneuvers being executed are shown, one representing a turn against the wind and another in favor of the wind, respectively on the left and right. The drawings represent present vessel heading, sail position, rudder position, apparent wind direction, tilting and time lapse stated.

As seen above, a sailboat can be effectively controlled by two independent Sugeno type fuzzy inference systems. This reflects common sailing practice where different persons act more or less independently on rudder and sail control. The system presented allows to keep the vessel on a predefined course and ensures a suitable sail position. The test runs presented did not show any conflict between the two fuzzy systems executed in parallel. The combination of the weather routing system and the described fuzzy control system allows the sailboat to reach any target completely autonomously.

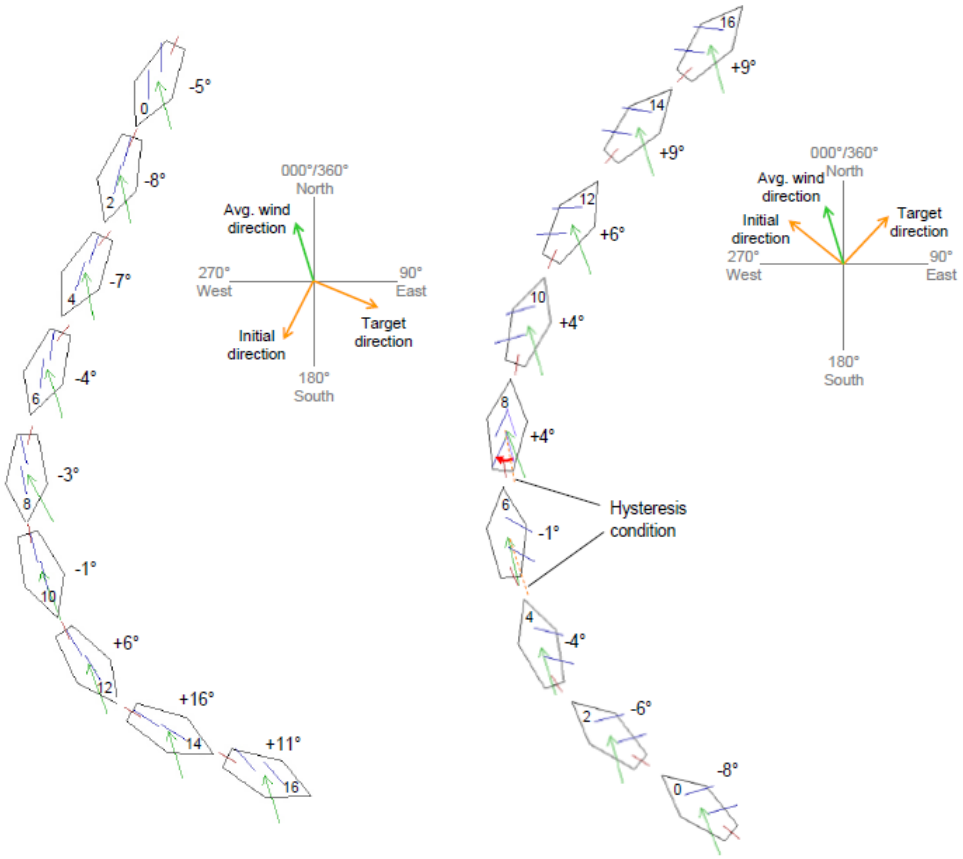


Figure 2.9: Test runs for turning against and in favor of the wind respectively with 2 second time intervals.

3

Supporting Concepts

This chapter introduces the reader to some of the key aspects of this dissertation, such as basic sailing theory, which is important to learn how to control the vessel. It is also important to know some basic aspects of fuzzy control theory in order to design a good controller.

3.1 Basics of Sailing

In order to design a good controller for a sailboat, it is necessary to know how to sail one properly. Also, it is important to know some essential characteristics of how the vessel is built, what is the usefulness of the most important features. Finally, basic sailing techniques are going to be viewed [Sal96]. In regard to sailing techniques, the veracity of the information was confirmed using the simulator Virtual Skipper 5, available on [Ent].

3.1.1 The vessel

The most common type of small to midsize sailboat is the sloop. The basic equipment needed for sailing is composed of one mast and two sails. The mainsail is usually a tall triangular sail mounted to the mast, with its foot along the boom, which extends towards the back from the mast. The sail in front, called the headsail or the jib, mounts on the forestay between the bow (frontal part of a ship) and the masthead, with its trailing corner controlled by the jibsheet. A general idea of these components position and sides of the ship are shown in figure 3.1.

The ship used for this project is a Killer Whales Radio Controlled Yacht, shown in figure 3.2. The user is given control of both the rudder direction and the sail trimming

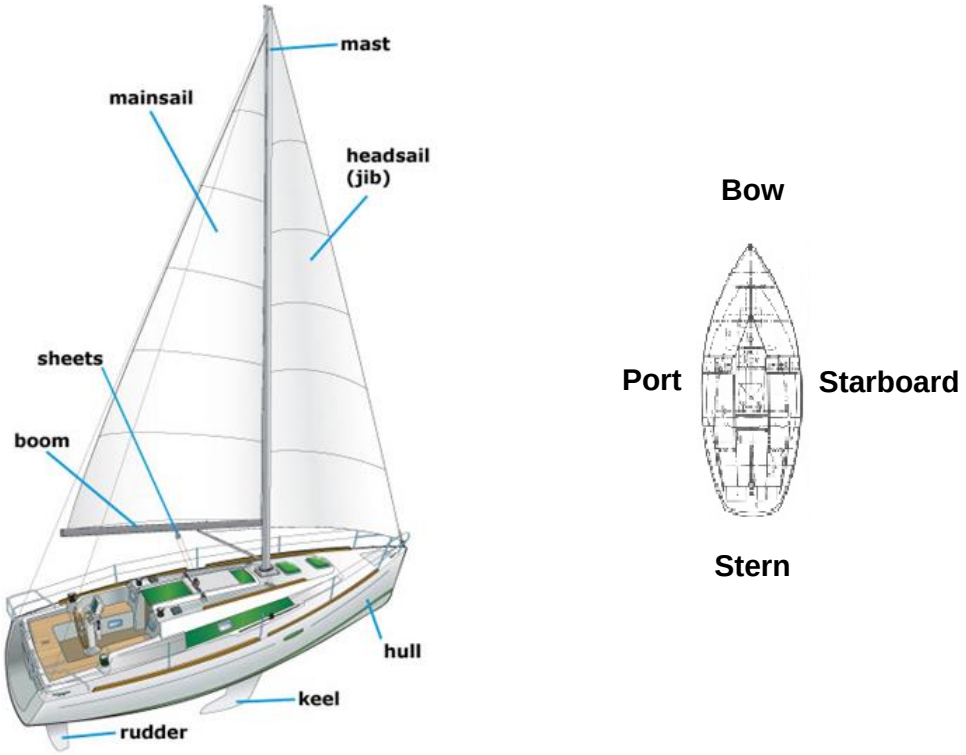


Figure 3.1: Main sailboat equipment [Boa] and side terms, viewed from above.

(tightening and loosening of the sail) of both mainsail and jib. Since this is a simple setup, only these two components and the keel are going to be analyzed. The sail will be analyzed in section 3.1.2.

Keel or Centreboard

If the wind is coming from behind the stern, the ship is pushed by it, but if the wind is coming from the side it pushes the vessel sideways. To reduce this effect, a boat needs a suitable underwater surface to lessen this drift. This drift, know as leeway, cannot be completely eliminated but can be kept within reasonable limits by the surface presented. It can also prevent capsizing. This device is the ship's keel. The two main categories of keels are swinging centreboards and daggerboards. Centreboards are a rotating type of keel that can be adjusted while sailing. Daggerboards are fixed boards inserted into a case, in the center of the vessel. The used vessel uses a daggerboard type keel.

Rudder

In contrast to cars, which follow in the direction set by the front wheels when steering is applied, a boat is steered by making its stern swing from one side to another. This can be achieved using a rudder, a device composed mainly of an underwater blade. The difference in pressure between the two sides of the blade pushes the stern sideways, thus moving the whole ship from one side to another.

Figure 3.2: Killer Whales Yacht.

3.1.2 The use of sails

Wind variables

The wind actually experienced on a vessel is called apparent wind. It is slightly different in strength and direction from the real (true) wind because a moving boat creates its own airflow, called the head wind. Apparent wind is a combination of head and true wind, as shown by figure 3.3.

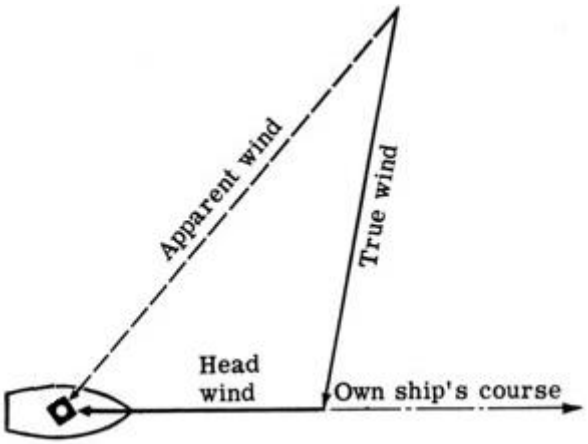


Figure 3.3: Experienced winds [Ass].

If a gust comes, true wind gets stronger. Since the response time of a vessel's speed

is not immediate, for a moment the head wind remains just as it was. This causes a momentarily change of direction of the apparent wind, making it come more from the side. On the other hand, if the windspeed drops suddenly the vessel's speed will me maintained for a few seconds and, following the same logic, the apparent wind will come more from ahead. On both situations the vessel's direction needs to be adjusted in order to prevent a loss of speed.

Sails as wind resistance

The earliest form of sailing was based on using the wind simply to push the boat along. The sail area was arranged as much at right-angles to the wind as possible. The more area available on the sail, the greater the resistance offered to the wind, thus more power was available for propulsion.

With the wind dead behind, the shape of the sail or sails is very important. The more concave or parachute like the shape of the sail the greater is the resistance offered to the wind. Yet, it is a mistake to think that a boat is sailing at its fastest downwind (with the wind dead behind). In this situation, the wind meets the sails at right-angles and eddies around behind them. This way of operation is more the exception than the rule. Most of the time the wind comes sideways and flows parallel to the curve of the sails, as such, they act as aerofoils.

Sails as aerofoils

The force of the wind can also be used to make progress against it. For this, the sail surfaces need to have an aerodynamic shape. In a cross-section view, sails resemble aircraft wings. When they are trimmed correctly the wind is deflected so delicately that the airflow remains unbroken. Lift (a combination of suction on one side and pressure on the other) is developed in the side furthest from the wind and acts at right-angles to the sail as shown in figure 3.4. Another smaller force acts parallel to the sails, caused by resistance offered to the airflow.

The force of the wind against the sails consists of two components, lift and resistance. The two can be expressed as a single force operating in between the two components, by means of a vector diagram. This force would be neither straight ahead nor at right-angles to the boat, it would act diagonally instead.

Therefore, the vessel is being pushed not only forward but also sideways. This sideways force is counteracted to a great extent by the keel, explained in subsection 3.1.1.

3.1.3 Sailing Techniques

Running before the wind

Sailing with the wind coming from the stern is the most basic form of sailing. The more resistance offered to the wind, in the form of a larger sail area, the faster the vessel goes.

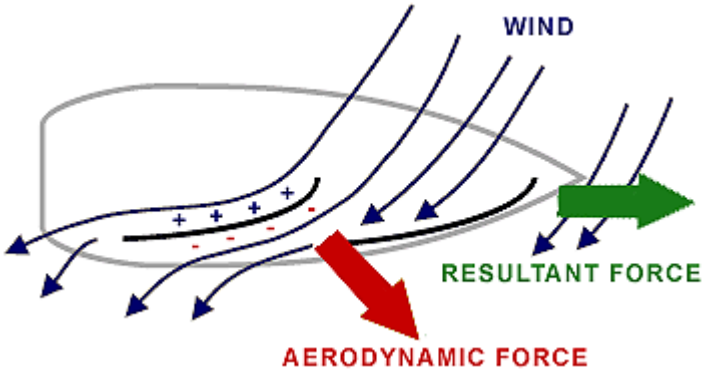
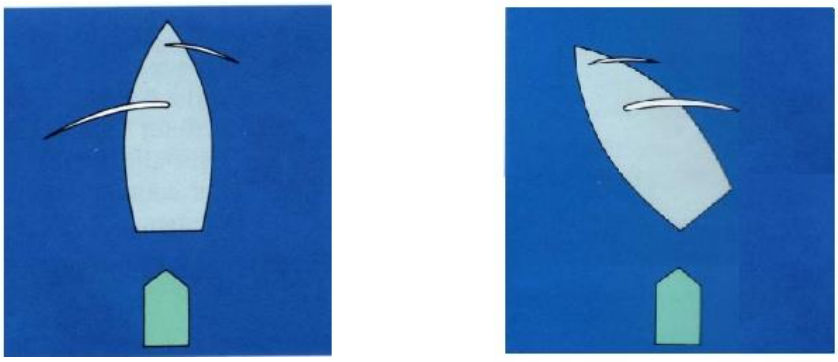


Figure 3.4: Lift and aerodynamic force created by the wind [Saib].

In modern sailing boats, this means arranging all the sails in such a way that the largest area possible is presented to the wind. For this to happen, the headsail and mainsail need to be on different sides (goosewinged) as seen on figure 3.5 a).

Broad reaching

If the wind is neither coming from the stern nor blowing at right-angles but coming from between these two, the vessel is sailing on a broad reach. Both sails are set on the farthest side from the wind (leeward) and the sheets are eased. This happens because the wind is not just striking the sails at right-angles but is also being deflected slightly by them. This makes the vessel go faster than running before the wind (figure 3.5 b)).



a) Running before the wind b) Broad reaching

Figure 3.5: Sailing techniques 1 [Sal96].

Bearing away

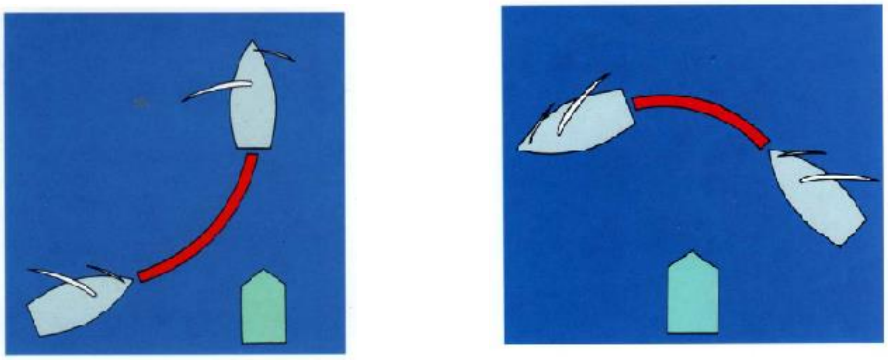
Altering the course so that the vessel turns away from the wind is defined as bearing away as seen on 3.6 a). As it is turning. the angle at which it meets the wind also changes, so the sails need to be adjusted progressively to suit the new situation. Since the wind is coming more and more from behind, the sail sheets are eased little by little.

Luffing

Luffing means altering the course towards the direction from which the wind is coming. It does not matter whether the vessel turns to port or starboard, luffing is always done when turning towards the wind. It is not simply a matter of steering, the sail sheets need to be adjusted progressively. They need to be gradually pulled in so that the vessel does not slow down, consequence of flapping sails.

If luffing is done from the stern to a beam reach (when the wind is roughly at right-angles to the vessel) the sails need to be tightened until they stop fluttering. If the wind is light the vessel should be allowed to heel so that gravity helps the sails adopt the correct angle and curvature.

If luffing is done until the vessel is close-hauled (sailing as close to the wind direction as possible without stopping, figure 3.6 b)) the sail sheets are progressively pulled in.



a) Bearing away b) Luffing

Figure 3.6: Sailing techniques 2 [Sal96].

Beating to windward

It is impossible to sail straight to the wind, instead, to beat to windward, or upwind, a zigzag route must be adopted as shown in figure 3.7 a). The aim is to sail fast without reaching too far from the objective. Short routes lead to too much loss of speed because the boat is heading too much into the wind to sail well. Also, if the boat does not point high enough into the wind the route becomes much longer than expected. A compromise between these two must be found.

It can be very difficult in practice to pick the best route to windward as windshifts also have to be taken into account.

Tacking

Tacking is defined as changing course of about 90^0 during which the sail crosses over from one side to the other. It involves entering and leaving the upwind zone, where it is impossible to sail because the wind is coming from dead ahead. As a result, it involves

a loss of speed and if the vessel does not have enough to carry through the turn it may not respond to the rudder and stop. The rudder should also not be turned too sharply because this too has a breaking effect. This maneuver is done at the end of each diagonal leg of a windward beating course.

Because the vessel is constantly losing speed doing this maneuver, it is important to get the wind back in the sails as quickly as possible and regain speed. Also as said before, if the rudder is turned too much it will act as a brake because the blade is at a sharp angle to the water flow. Another compromise has to be found between loss of speed and time taken to do the tacking maneuver. If the rudder is not pushed far enough, there is a long delay with constant speed dropping until the wind catches the sails again. It is generally accepted that the best rudder angle is about 33^0 to the centreline of the boat. It is specially important not to come to a full stop while tacking because a sailing boat without speed cannot be steered.

Stopping

The only way to stop a sailing boat is to turn it directly into the wind to make it start losing speed. Judgement is needed to gauge how far the vessel will travel head-to-wind until it stops. Once the goal is reached, the sail sheets are fully released and the rudder is turned hard into the direction of the wind as shown in figure 3.7 b).

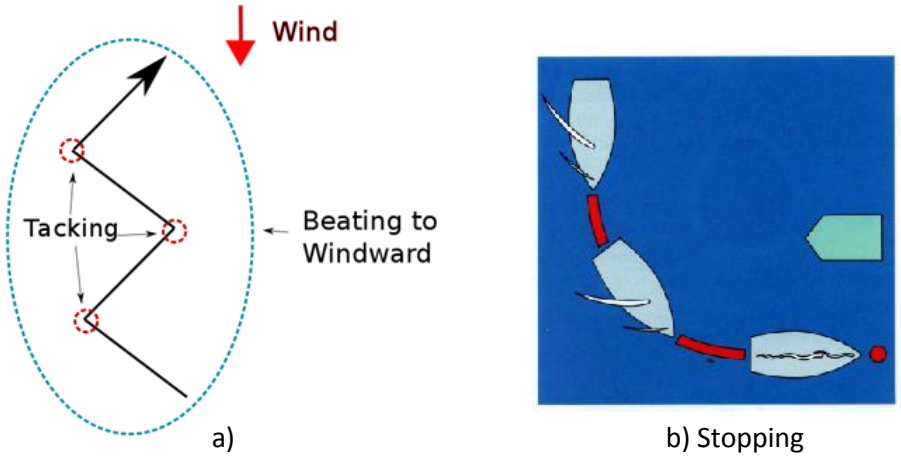


Figure 3.7: Sailing techniques 3 [Cat, Sal96].

3.2 Fuzzy Control

3.2.1 An Introduction

Knowledge representation is one of the most actively research areas in Artificial Intelligence [LB87, Moo82, Neg85]. Still, there are many important issues in this area that have not been adequately solved. One of this issues is the representation of imprecise knowledge.

Conventional knowledge representation techniques do not provide effective tools for representing the meaning of everyday like facts such as:

- *Usually* it takes *about an hour* to drive from Berkeley to Stanford in *light* traffic.
- *Most* experts believe that the likelihood of a *severe* earthquake in the *near* future is *very low*.

The words in italic in the assertions are labels for fuzzy predicates, quantifiers and probabilities. Conventional approaches to knowledge representation lack the means to represent fuzzy concepts. Consequently, approaches based in first order logic and classical probability theory do not provide an appropriate framework for dealing with commonsense knowledge since such knowledge is both imprecise and noncategorical by nature [Zad89].

The development of fuzzy logic was motivated mainly by the need for a conceptual framework capable of addressing the issues of uncertainty and lexical imprecision. Nowadays fuzzy logic has emerged as an alternative to classical logic in application areas ranging from industrial process control to aerospace and bioengineering [Sug85, Zim91].

3.2.2 Applications

Possibly the most impressive fact about the present success of fuzzy logic is the wide array of applications of this paradigm, ranging from consumer products to automotive engineering. Table 3.1 illustrates this point with a list of fuzzy logic uses in an industrial setting. In brief, fuzzy logic plays a similarly central role in shaping a suitable rule-based or linguistic control strategy in these applications.

Furthermore, in the majority of these cases, fuzzy logic bridges the gap between symbolic processing and numeric computation, thereby expanding the domain of application of control engineering to those outside its realm of usage. Specifically, fuzzy logic can form the basis of implementation of control strategies in the wide sense to include decision making and supervisory control.

Table 3.1: Applications of fuzzy logic control, as seen in [YLZ95] page 6.

Consumer Products	Automotive and Power Generation	Industrial Process Control	Robotics and Manufacturing
•Cameras and camcorders (Canon, Minolta, Ricoh, Sanyo) •Washing machines (AEG, Sharp, Goldstar) •Refrigerators (Whirlpool)	•Power train and transmission control (GM-Saturn, Honda, Mazda) •Engine control (Nissan)	•Cement kiln, inceneration plant (K.L.Smith) •Refining, distillation and other chemical processes	•Electrical discharge machine (Mitsubishi)

3.2.3 Essential characteristics

As the name suggests, fuzzy logic is the logic underlying modes of reasoning which are approximate, and not entirely exact. The importance of fuzzy logic comes from the fact that most modes of human reasoning are approximate in nature, such as commonsense reasoning. It is interesting to note that approximate reasoning falls outside the range of classic logic, because the main concern of the later topic is the reasoning that leads to precise formulation and analysis.

Some of the main characteristics of fuzzy logic relate to the following:

- Exact reasoning is viewed as a limited case of approximate reasoning;
- Everything is a matter of degree;
- Any logical system can be fuzzified;
- Inference is viewed as a process of propagation of flexible constraints;

Fuzzy logic differs from traditional logical systems in spirit and detail. Some of the main differences are summarized in the following [Zad83].

- *Truth:* In bivalent logical systems, truth can have two values: *true* or *false*. In multi-valued systems it can be an element of a finite set, an interval or a boolean algebra equation. In fuzzy logic, truth may be a fuzzy subset of any partially ordered set or, a point in the interval $[0, 1]$. The so-called linguistic truth values such as *true*, *partially true*, *not quite true* are interpreted as labels of fuzzy subsets on an unit interval.
- *Predicates:* In bivalent systems, predicates are crisp, like *mortal*, *even*, *larger than*. In fuzzy logic, predicates are fuzzy, such as *tall*, *ill*, *soon*, *much larger than*. It should be noted that most predicates in natural language are fuzzy, and not crisp.
- *Predicate Modifiers:* In classic systems, the most widely used predicate modifier is the negation (*not*). In fuzzy logic there is a variety of predicate modifiers which act as mitigators or weakeners of statements such as *very*, *more or less*, *quite*, *rather*, *much*. Such predicate modifiers play an essential role in the generation of the values of a linguistic variable : *very young*, *more or less young*, *rather young*, etc [Zad73].
- *Quantifiers:* In classical logical systems, only two quantifiers exist, universal and existential. Fuzzy logic admits a wide variety of fuzzy quantifiers exemplified by *few*, *several*, *usually*, *most*, *always*, *frequently*, etc.
- *Probabilities:* In classical logical systems, probability is numerical or interval valued. In fuzzy logic there is also the additional option of linguistic probabilities, exemplified by *likely*, *unlikely*, *around 0.5*, *high*, etc [Zad86].

It is important to notice that in every instance fuzzy logic adds to the options available in classical logical systems. Fuzzy logic can then be viewed as an extension of these systems, rather than a system of reasoning in conflict with classical systems.

3.2.4 Fuzzy logic controller architecture

Different methods for developing fuzzy logic controllers have been proposed. In the design of a fuzzy controller, one must identify the main control parameters and determine a term set which is at the right level of specification for describing the values of each linguistic variable. For instance, a term set including linguistic variables such as {*Small, Medium, Large*} may not be enough to satisfy some domains, it may be needed to have more terms in the set, such as {*Very Small, Small, Medium, Large, Very Large*} [Ber92].

Figure 3.8 illustrates a basic architecture for a fuzzy logic controller. This architecture consists on four modules whose functions will be described.

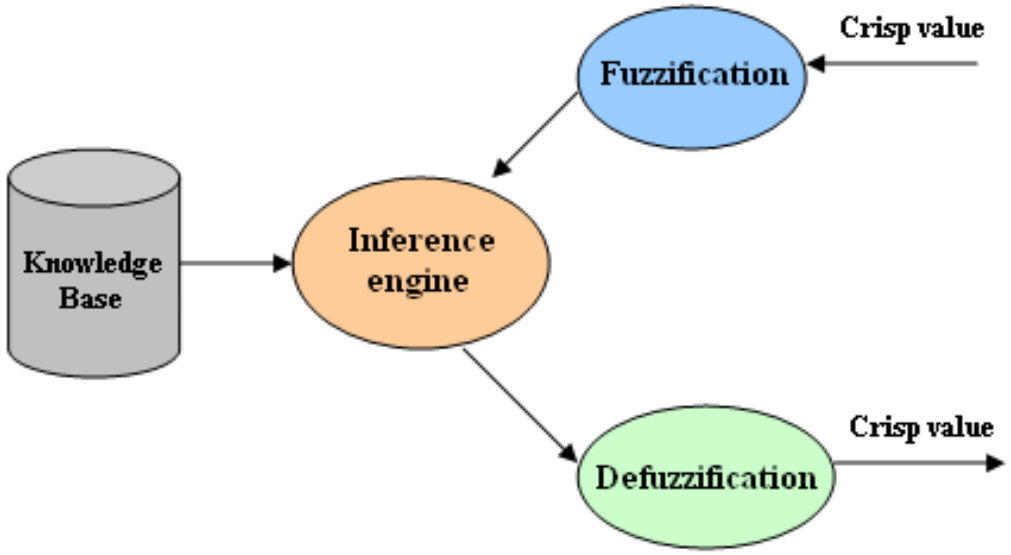


Figure 3.8: Fuzzy logic controller [Alo].

Fuzzy Sets

A fuzzy set is an extension of a crisp set. A fuzzy set, defined over some domain of definition, is a mapping from that domain into the interval $[0, 1]$ This mapping is called the membership function of a given fuzzy set. It takes the value of zero for no membership and one for full membership. In crisp sets these are the only two allowed values. Fuzzy sets also allow partial, or graded membership. In other words, an element may partially belong to a fuzzy set. Different types of fuzzy membership functions have been used to define fuzzy sets. Some of them are shown in figure 3.9 [Ber92].

The first and second types use triangular (a) and trapezoidal (b) membership functions. Due the simplicity of the formula and computational efficiency, these membership functions are extensively used, especially in real-time implementations. However, since they are composed of straight line segments, there is no smoothness on the corner points specified by the parameters. To cope with this, bell-shaped functions can be used (c). Another used type, specially in Sugeno-style inference is the fuzzy singleton (d). This

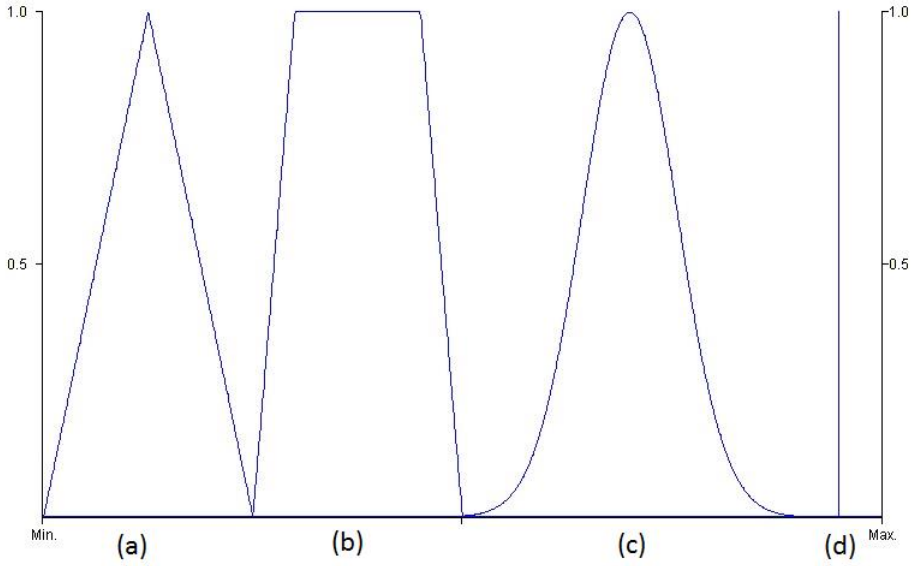


Figure 3.9: Different membership function types: (a) - Triangular ; (b) - Trapezoidal; (c) - Bell-shaped; (d) - Singleton;

membership function is described as unity at a single particular point on the universe of discourse, and zero everywhere else. The use of singletons is directly tied to the computational effectiveness of a fuzzy controller.

Fuzzification strategy

In coding the values from outside the fuzzy system, one transforms the given values in terms of the linguistic variables used in the preconditions of the ruleset. If the given value is crisp, then the fuzzification stage requires matching the measurement against the membership function of the linguistic label as shown in figure 3.10. If the given value contains noise, it may be modeled using a triangular membership function where the peak of the triangle refers to the mean value of the data set and the base refers to a function of the standard deviation. The most widely used fuzzification method is the former case when the given value is crisp [Ber92].

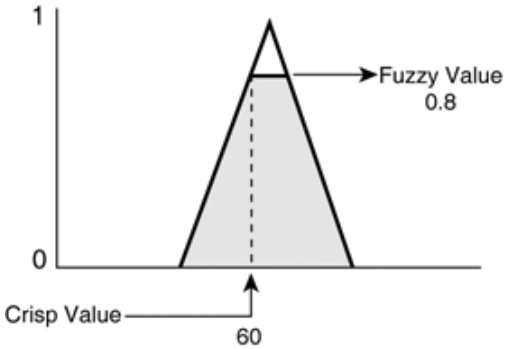


Figure 3.10: Fuzzification stage for crisp inputs [Yal].

Knowledge/rule base setup

There are two important tasks in designing the knowledge/rule base setup. First, a set of linguistic variables must be properly selected in order to describe the main control parameters of the system. Both the input and output parameters must be linguistically defined using proper term sets. Also, the granularity of a term set is important for the smoothness of control. Secondly, a control knowledge base must be developed which uses the linguistic description described in the first point. Sugeno [Sug85] suggested four methods for doing this:

1. Expert's knowledge
2. Modeling the operator's control actions
3. Modeling a process
4. Self organization

Among these methods, the first method is the most used [MA75]. This method is effective when expert human operators can express the heuristics that they use in controlling a process in terms of rules.

The second method, directly models the control actions of the operator. Instead of interviewing the operator, its control actions are imitated by the system.

The third method deals with fuzzy modeling of a process where an approximate model is configured by using implications describing possible states of a system. In this method, a model is developed and a fuzzy controller is constructed to control the fuzzy model, making this approach very similar to traditional approaches in control theory.

The fourth method's main idea is to develop rules which can be adjusted over time to improve the controller's performance.

These rules are a collection of linguistic statements that describe how the system should make a decision regarding classification of inputs and control of outputs. Fuzzy rules are always written in the following form:

if *(input1 is membership function1)* **and/or** (*input2 is membership function2)* **and/or** ... **then** *(outputX is output membership functionX)*.

Inference process

In a fuzzy ruleset, there is the possibility of more than one rule being fired at the same time, because of the partial matching attribute of fuzzy control rules and the fact that the preconditions of the rules can overlap each other. The methodology used in deciding what control action should be taken as the result of multiple rules being fired can be referred to as the process of conflict resolution, or inference.

In a two input, two rule inference system shown in figure 3.11 the process occurs in the following way:

1. The strength for each rule is calculated by taking the minimum of the two input membership values (boolean "and").
2. The control output of each rule is calculated by applying the matching strength of the rule on its conclusion.
3. The control outputs of all fuzzy rules are combined to obtain one fuzzy output distribution. This output usually is obtained using a boolean "or". This technique combines the fuzzy membership functions into one.

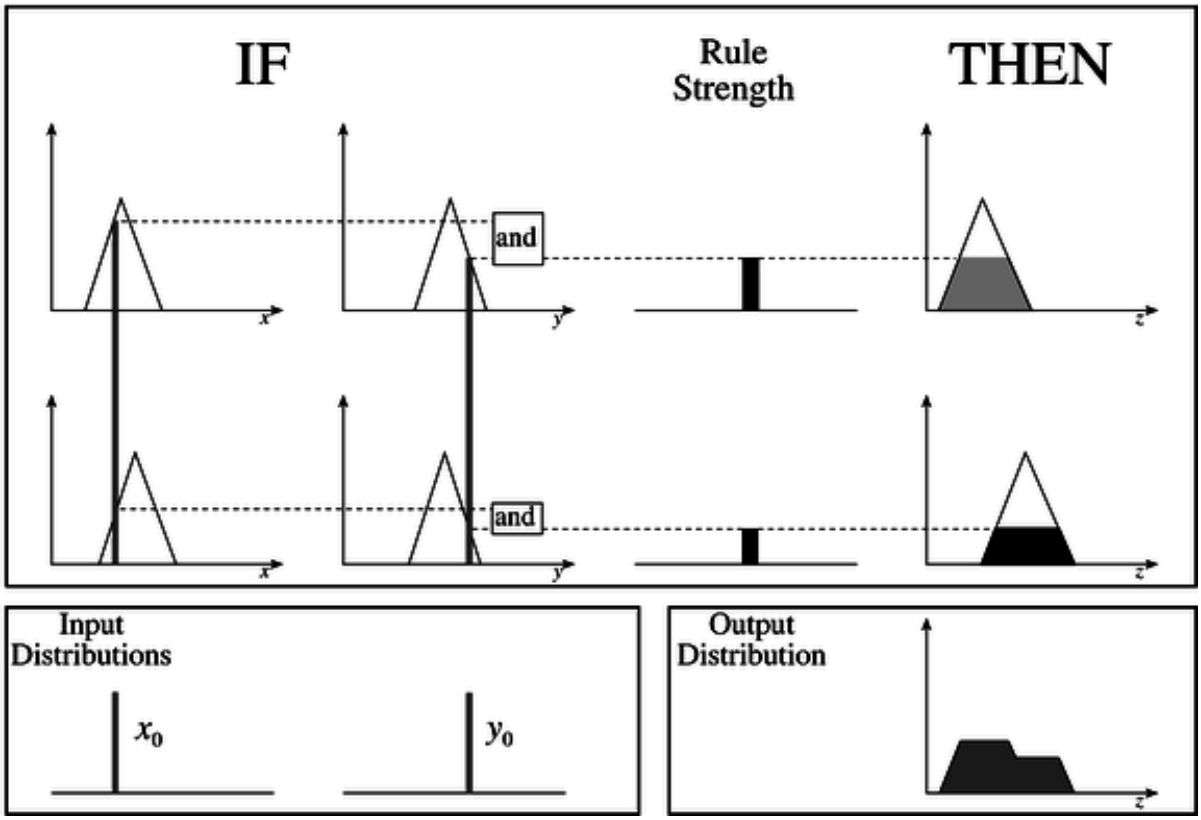


Figure 3.11: Mamdani two input, two rule inference system [Kna].

Mamdani-type inference, as shown above, expects the output membership functions to be fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that will need defuzzification. It is possible and much more efficient to use a singleton membership function than a distributed fuzzy set. Singletons, used in Sugeno-type systems, can be thought of as pre-defuzzified fuzzy sets.

The result of this operation is a membership function and has to be defuzzified into a single crisp value in order to be used.

Defuzzification strategy

The last operation produces a nonfuzzy control action that best represents the membership function of an inferred control action. Several defuzzification strategies have been

suggested in literature. Among them, three of the more often applied methods are described [Ber92].

Tsukamoto's defuzzification method calculates a crisp control action, if a monotonic membership function is used as shown in equation 3.1. This method is also known as weighted average method and is generally used in Sugeno-type inference systems. n is the number of rules with firing strength (ω_i) greater than 0 and x_i is the amount of control action recommended by rule i.

$$Z^* = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} \tag{3.1}$$

The center of area method calculates the center of gravity of the distribution for the control action, assuming that it has has been produced with a point wise membership function. Assuming a discrete universe of discourse we have equation 3.2 where q is the number of quantization levels of the output, z_j is the amount of control outputs at the quantization level j and $\mu_C(z_j)$ represents its membership value in C.

$$Z^* = \frac{\sum_{j=1}^{q} z_j \mu_C(z_j)}{\sum_{j=1}^{q} \mu_C(z_j)} \tag{3.2}$$

Finally, the mean of maximum method generates a crisp control action by averaging the support values which their membership values reach the maximum. For a discrete universe, it is calculated following equation 3.3 where l is the number of quantized z values which reach their maximum memberships.

$$Z^* = \sum_{j=1}^{l} \frac{z_j}{l} \tag{3.3}$$

4

Software and Hardware

This chapter introduces the technologies used in the elaboration of this dissertation, namely the prototyping platform, sensors and software used for model construction / system emulation.

4.1 Arduino

Arduino is an open-source electronics prototyping platform based on flexible hardware and software. Its original purpose is the development of interactive projects. The platform can sense the environment around it by receiving inputs from a variety of sensors and can affect its surroundings by controlling a series of actuators, such as motors. The microcontroller on the board is programmed using Wiring [BHB] based programming language and the development environment is based on Processing [FR].

There are many microcontroller platforms available for physical computing, such as Parallax Basic Stamp, Netmedia's BX-24 and MIT's Handyboard. All of these tools are designed to transform the complicated issue of microcontroller programming into an easy to use package. Arduino also has the same functionalities but it also offers some advantages over other systems such as:

- Cost-effectiveness: Arduino boards are relatively inexpensive compared to other microcontroller platforms. The simplest versions of the Arduino module can be assembled by hand and even pre-assembled modules cost less than 20 Euros.

- Cross-platform: Arduino software runs on Windows, Mac-OS and Linux operating systems. Most microcontroller development systems are limited to Windows.

- Open source and extensible software: The Arduino software is published as open source tools, available for expansion by experienced programmers. The language can be expanded through C++ libraries and by adding AVR-C code directly into Arduino programs.

- Open source and extensible hardware: Arduino is based on ATMEGA microcontroller family. The plans for the modules are published under a special license that enables experienced designers to make their own versions of the module, extending and further improving it.

4.1.1 Hardware

The Arduino board used for this project is an Arduino Mega 2560 [Ban] as shown in picture 4.1.

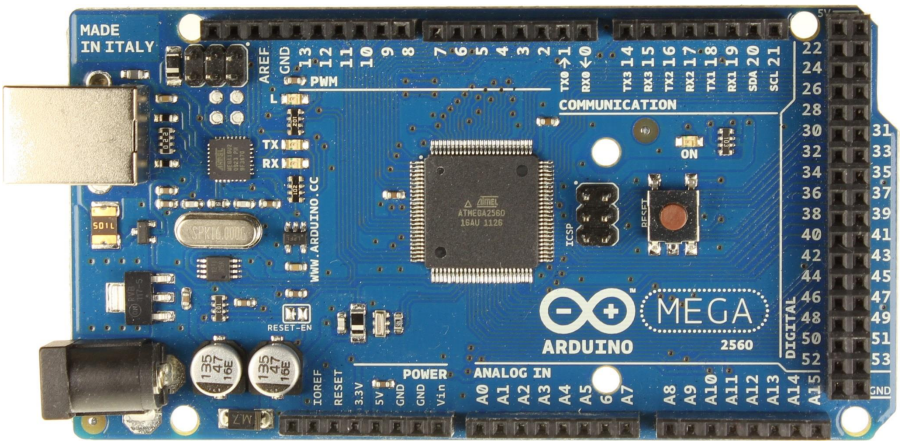


Figure 4.1: Arduino Mega 2560 board [Ban].

This is a microcontroller board based on the ATmega2560 [Cor12] with characteristics such as 4 hardware serial ports, USB connection, reset button, ICSP (In Circuit Serial Programming) header and others. The main characteristics of this board are listed in table 4.1.

In terms of power supply, Arduino Mega can be powered via USB connection or with an external power supply. The source is selected automatically. With an external supply the board can operate on the $6 - 20$ volt range. However, less than 7V supply can turn the board unstable and more than 12V may damage the board, as such the recommended range of operation is 7 to 12 volt.

The 54 digital pins can be used as inputs or outputs, in addition, some pins have specialized functions:

- Serial 0: 0(RX), 1(TX); Serial 1: 19(RX), 18(TX); Serial 2: 17(RX), 16(TX); Serial 3: 15(RX) and 14(TX). Used to receive (RX) and transmit (TX) TTL serial data.

- External Interrupts: 2(Interrupt 0), 3(Interrupt 1), 21(Interrupt 2), 20 (Interrupt 3),

Table 4.1: Arduino Mega 2560 features.

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

19 (interrupt 4), 18 (interrupt 5). These pins can be configured to trigger an interrupt on a low value, rising edge, falling edge or a change in value.

- PWM: 2–13 ; 44–46. Provides 8-bit PWM output.
- LED: 13. There is a built-in LED connected to digital pin 13 which is ON if the pin is HIGH value and OFF when the pin is LOW.
- SPI: SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI (Serial Peripheral Interface) using a library created for the effect. These pins are also broken out on the ICSP header, which is physically compatible with other Arduino boards.
- TWI: 20 (SDA) and 21 (SCL). Supports TWI(Two-wire interface) using a library created for the effect.

The 16 analog inputs provide 10 bits of resolution (1024 different values). By default they measure from ground to 5 volts though it is possible to change the upper end of their range.

4.2 Proposed sensors for the project

The initial set of sensors included a wind vane (to measure wind direction), anemometer (to measure wind speed), compass (to know the vessel's heading and inclination) and GPS (to obtain the vessel's position and speed). The proposed models are shown in this section.

4.2.1 Weather meter

The core components of weather measurement are wind speed, wind direction and rainfall. These components are present in the kit available at [Sysb], shown in figure 4.2a) and two of them are necessary for the project at hand.

None of the sensors in this kit contain active electronics, instead they use sealed magnetic switches and magnets . The cup anemometer encodes the wind speed by closing a switch with each rotation, A windspeed of 1,492 Miles per Hour produces a switch closure once per second. The wind vane reports wind direction as a voltage produced by a combination of resistors inside the sensor. When a voltage is supplied, the return can be translated to any of 16 possible positions. The users manual has a table of voltage and resistance values for each of the 16 positions [Sysa].

4.2.2 CMPS10 Tilt Compensated Magnetic Compass

The CMPS10 module shown in figure 4.2 b) is a tilt compensated compass employing a 3-axis magnetometer and a 3-axis accelerometer and a 16-bit processor, it has been designed to remove the errors caused by tilting of the PCB (Printed Circuit Board). The CMPS10 produces a result of 0-3599 representing 0-359.9 or 0 to 255. The output of the three sensors measuring x, y and z components of the magnetic field, together with the pitch and roll are used to calculate the bearing, each of these components are also made individually available [PTRb].

The module requires a power supply at 3.3 - 5v and draws a nominal 25mA of current. There are three ways of getting the bearing from the module: A serial interface, an I2C interface or a PWM output. More information about this sensor is available in [Ele].

4.2.3 EM-406A SiRF III GPS Receiver with Antenna

The EM-406A GPS module shown in figure 4.2 c) includes on-board voltage regulation, LED status indicator, battery backed RAM, and a built-in patch antenna. Its main features are a 20 channel receiver, high sensitivity (-159dBm) and 10m positional accuracy [PTRa].

This module requires a power supply of 4.5 - 5.5v and draws a nominal 70mA of current. It has an average cold starting time of 42 seconds and an average 1 second of hot start. For more information on this sensor check [Glo].

It is also important to note the protocol used by GPS sensors, the NMEA (National Marine Electronics Association) protocol [Sta]. This protocol defines the interface between various pieces of marine electronic equipment and permits information sending from marine equipment to computers and vice versa. Standard GPS receiver communication is defined on the NMEA specification.

The idea of NMEA is to send a line of data (an NMEA sentence) that is independent from other NMEA sentences. There are standard sentences for each device category and the ability to define proprietary sentences. All standard sentences have a two letter prefix that identifies the device using the sentences (for GPS receivers the prefix is GP) followed by a three letter sequence that defines the sentence contents. Standard sentences are constructed using the following rules:

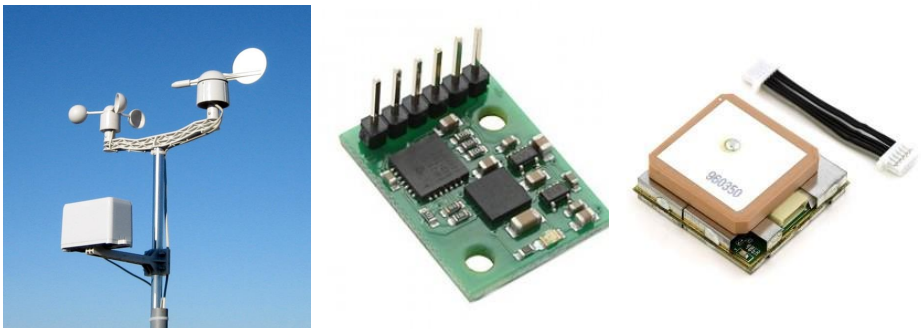
- Each sentence begins with the '$' symbol.

- The following five characters indicate the origin and type of message as explained.
- Different data items in the sentence are separated by commas.
- If no data is available in an item it is sent null, for example "..123,,456...".
- The last item is a checksum field, consisting of a '*' character and two hexadecimal digits representing an 8 bit exclusive OR of all characters between '$' and '*'. This checksum is obligatory on some sentences only.

The sentence used in this dissertation is the RMC (Recommended minimum data for GPS). This sentence has all the data needed for the controller's specification as it will be explained later. RMC data format is shown in table 4.2.

Table 4.2: RMC Data Format, adapted from [ST].

Name	Example	Units	Description
Message ID	$GPRMC		RMC protocol header
UTC time	161229.487		hhmmss.sss
Status	A		A=data valid or V=data not valid
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
Speed Over Ground	0.13	knots	
Course Over Ground	309.62	degrees	True
Date	120598		ddmmyy
Magnetic Variation	003.1	degrees	E=east or W=west
Mode	A		A=Autonomous, D=DGPS, E=DR
Checksum	*10		
<CR> <LF>			End of message termination



a) b) c)

Figure 4.2: Proposed sensors: a)Weather Meter [Sysb]; b)Compass Module [PTRb]; c)GPS Module [PTRa];

4.3 Available sensors and emulators

While the proposed sensors were unavailable, progress was mostly done using emulators and software to artificially recreate and simulate the needed environment. Using the same type of interfaces between the sensors and the Arduino prototyping board it is possible to test the behavior of developed algorithms. In order to achieve this, a kit using a PIC18F4550 microcontroller was developed [Gil13]. This emulator receives instructions via USB from a computer through a simulation platform and implements the original interface communication (original protocols used where UART [Mica] and SSI [NI]) from each sensor to the Arduino. To complete the cycle, Arduino implements the PWM signals needed to move the rudder and sail servomotors just as shown in figure 4.3. All the work presented in this section can be consulted in [Gil13].

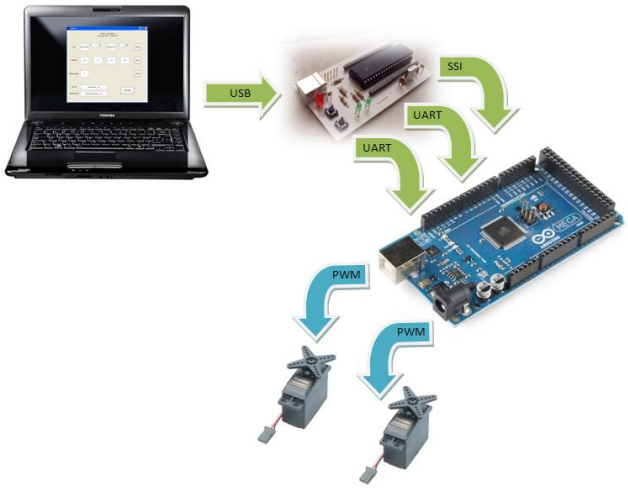


Figure 4.3: Sensorial emulation setup [Gil13].

4.3.1 Simulation platform

In order to obtain and easy to use interface a simulator was developed in MATLAB. The user simply has to fill forms and send information to the available emulators. This simulator operates on two different modes:

- Step-by-Step mode
- Automatic mode

On Step-by-Step mode, the user fills a form with the information he/she wants to send, referent to each sensor. Clicking a send button, the respective data is sent by USB to the PIC18F4550 through a virtual COM interface [Micb]. Each sensor is individually identified by the characteristics of the protocol it uses and the types of data sent.

On Automatic mode, the user only needs to indicate the location of previously built files with data from compass and GPS courses. This mode does not contemplate wind data. Both interfaces are shown in figure 4.4.

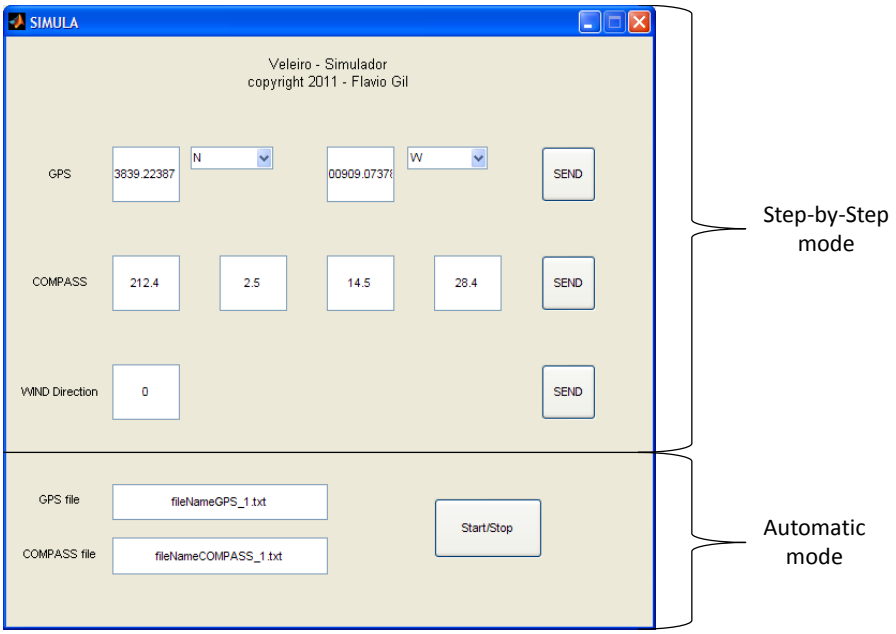


Figure 4.4: Simulation interface, adapted from [Gil13].

4.3.2 PIC-Arduino interface

The emulator created using a PIC18F4550 microcontroller in order to emulate sensorial data uses two different types of interfaces with the system: two UART interfaces and one SSI. Data receiving uses an USB connection from the computer, the sensor array use UART for the GPS and compass modules, and SSI for the wind direction sensor (originally, an AS5040 [Aus] sensor was going to be used for wind direction calculation).

The main algorithm implemented on this microcontroller behaves as a regulator that routes messages received via USB to their respective destinations. Message association is made through the first two characters present on the string message. If the first character is not '$' then the message is relative to the AS5040 sensor. If the first character is '$' then the message is relative to the GPS or compass modules, which are distinguished by the second character being 'G' or 'C', respectively. Besides data reception and transmission, the PIC18F4550 implements an interrupt routine capable of replying to the Arduino's solicitations.

4.4 Xfuzzy

Applying an available technology in another technology can sometimes be more than a simple task. In order to design a fuzzy logic controller usable via the Arduino programming language a dedicated development environment was used.The fuzzy system development environment Xfuzzy integrates a set of tools that ease the user's coverage over the several stages involved in the design process of fuzzy logic-based inference systems [VBSB03, MVBBSS].

The version used in the aiding of this project was Xfuzzy 3 [dMdSc]. Its main features are the capability for developing complex systems and the flexibility of allowing the user to extend the set of available functions. The environment has been completely programmed in Java, so it can be executed on any platform with JRE (Java Runtime Environment) installed. The design flow of Xfuzzy 3 includes the following stages:

1. Description stage - This stage includes graphical tools for the fuzzy system definition and development. This definition of the system is made using the xfedit tool [dMdSd]. This tool defines the linguistic variables, logical relations between them, operator sets and rule bases of the created system.

2. Verification stage - This stage includes tools for simulation, monitoring and representing graphically the system behavior. The aim is to detect possible deviations on the expected behavior and to identify the deviation sources. From the tools available for verification, the simplest to use is the xfmt tool [dMdSb]. This monitoring tool shows the activation degree of every linguistic label and logical rule, as well as the value of the different input variables, for a given set of input values.

3. Tuning stage - This stage consists in applying identification, learning and simplification algorithms to the fuzzy system. It is usually one of the most complex tasks when designing a fuzzy system. The system behavior depends on the logic structure of its rule base and the membership functions of its linguistic variables. The tuning process if often focused on adjusting membership function parameters. Manually this process is cumbersome, automatic techniques are preferable. The two most widely used learning mechanism and supervised and reinforcement learning. Xfuzzy 3 currently contains one tool dedicated to supervised learning algorithms. The tuning stage was not reached in this dissertation.

4. Synthesis stage - This stage includes tools for generating high level descriptions for software and hardware implementations. Software representations are useful when there are not strong restrictions on the inference speed, system size and power consumption. They can be generated from any fuzzy system developed in Xfuzzy without restraints. On the other hand hardware representations are more adequate when high speed, small area or low power consumption is needed. For this to happen some constraints must be imposed to the fuzzy system. Xfuzzy 3 provides three tools for software synthesis and two tools for hardware synthesis. For the purpose of this dissertation, the xfcpp [dMdSa] tool was used to develop a C++ description.

5

Proposed controller

This chapter will introduce all the work developed for this dissertation. Caused by the lack of sensors some methods could not be tested. Section 5.1 introduces a method for automatic control transition and the sections after explain the design used for the controller. The final design can be viewed in figure 5.1. Most of the protocols used and routines for data treatment are explained in detail in [Gil13].

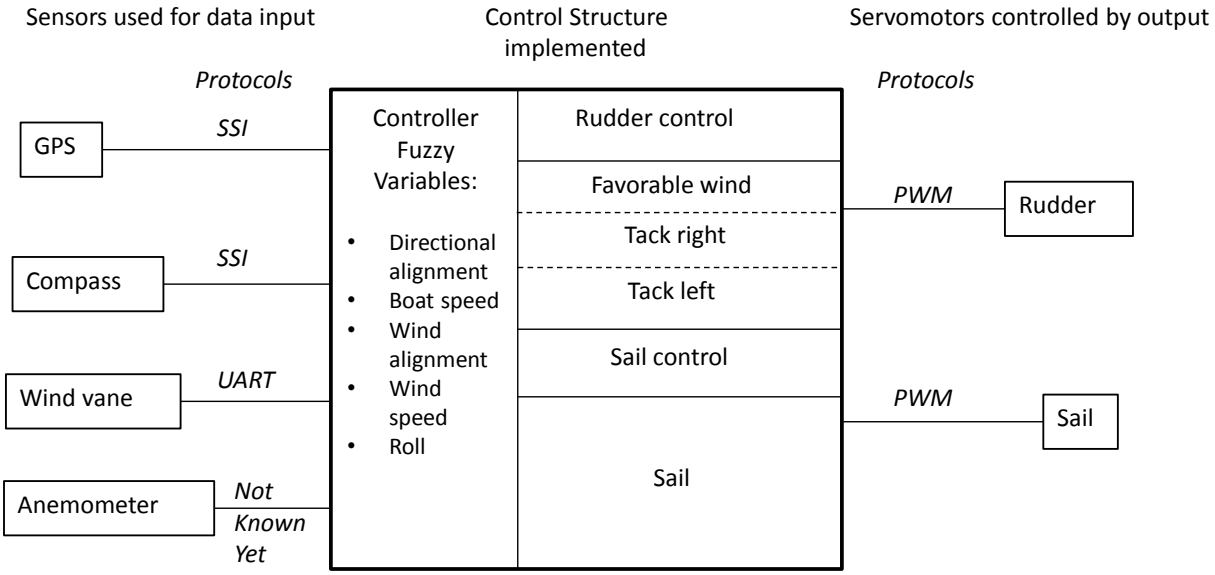


Figure 5.1: Signal generated when the remote controller is off.

5.1 Transition between manual and automatic control

Since a prototype of an automatic system is prone to failure, it is always good practice to add safety failsafe methods. On the prototype vessel used there is the possibility of using the original manual remote controller to add this specification. The objective is to give manual control to the user as soon as the remote controller is on, if the controller is off then the remote system controls the vessel.

The Killer Whales Yacht comes equipped with a Radio Frequency module for communicating with the remote controller. The signal generated from this module is different whether the remote controller is on or off, as shown in figures 5.2 and 5.3.

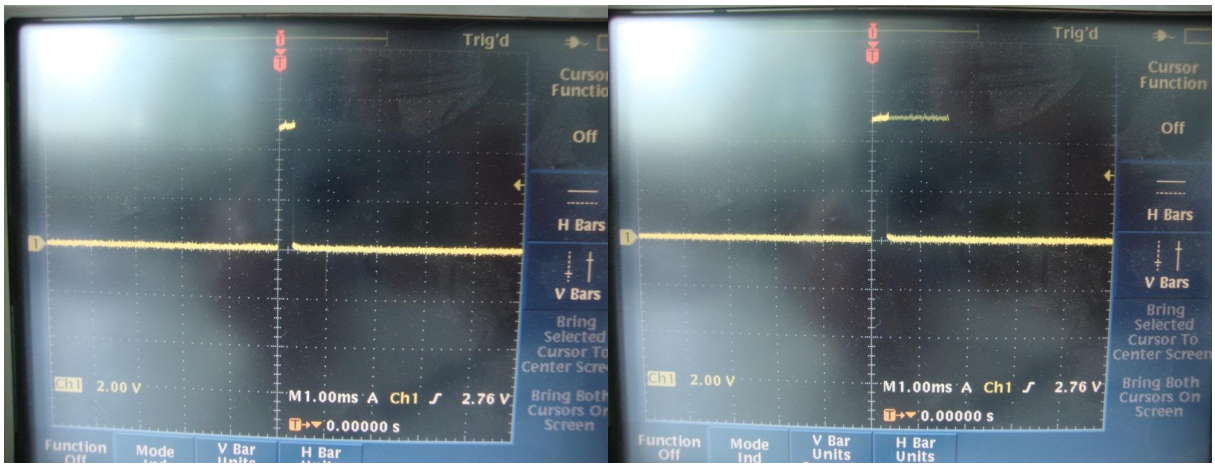


Figure 5.2: Signal generated when the remote controller is off.

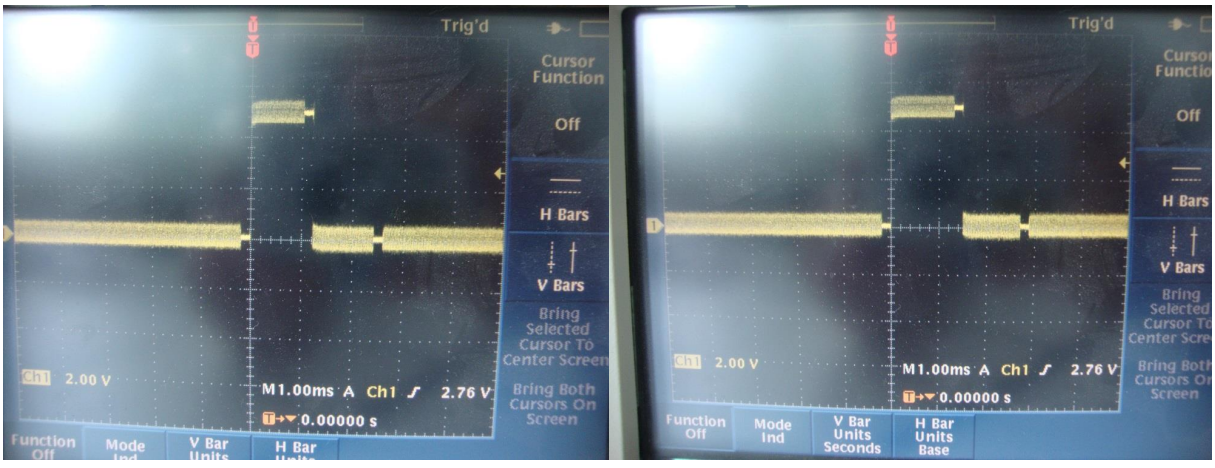


Figure 5.3: Signal generated when the remote controller is on (noise is irrelevant in this case as it is only generated from the RF waves into the oscilloscope).

Distinguishing both signals give the possibility to pass from remote to manual control and from manual to remote control, via multiplexing. The data inputs will be the original signal from the RF module (manual control) and the signal provided by the Arduino prototyping board (automatic control) as shown in figure 5.4.

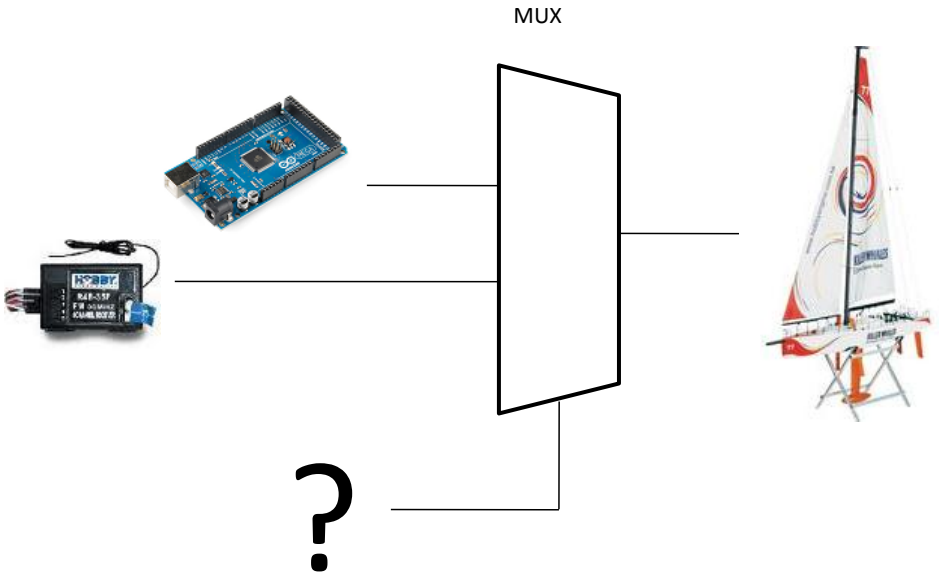
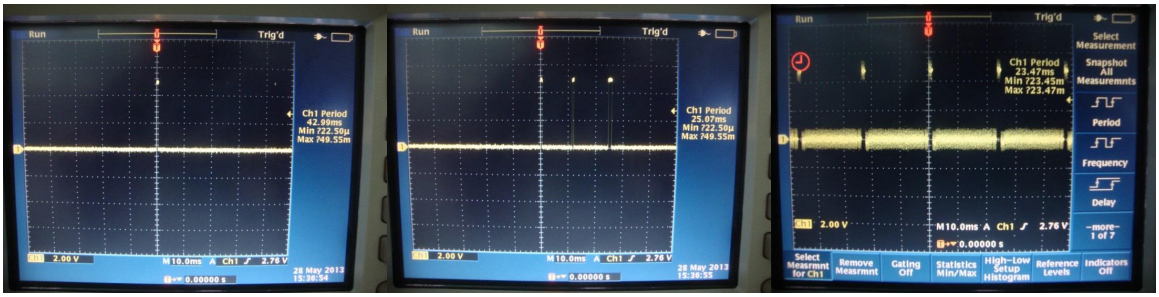


Figure 5.4: Transition between manual and remote control.

There is still the need to know how to select the right input to control the sailboat. To solve this problem, the PWM values associated with the rudder and sail servo-motors generated in the RF Module were used. Figure 5.5 shows two random signals with the controller off and one stable signal with the controller on.



Controller off Controller on

Figure 5.5: Different characteristics in the controller's signal.

After analysis it is easily seen that with the controller on, the RF module signal is a PWM pulse of $\pm 23ms$ and with the controller off this signal becomes a series of random pulses. Also, the total period of the positive signal (5V) is less when the controller is off. Using this property it is possible to distinguish the two types of signal through the method described in figure 5.6.

Using a counter it is possible to determine whether the controller is on or off (5.6 a). From Z *ms* to Z *ms* the RF module signal is checked upon, if it has a positive signal (1) the counter is incremented in X units, if not it is decremented in Y units. Since the period between pulses is quite superior to the total period of a single pulse the increment should also be superior to the decrement.

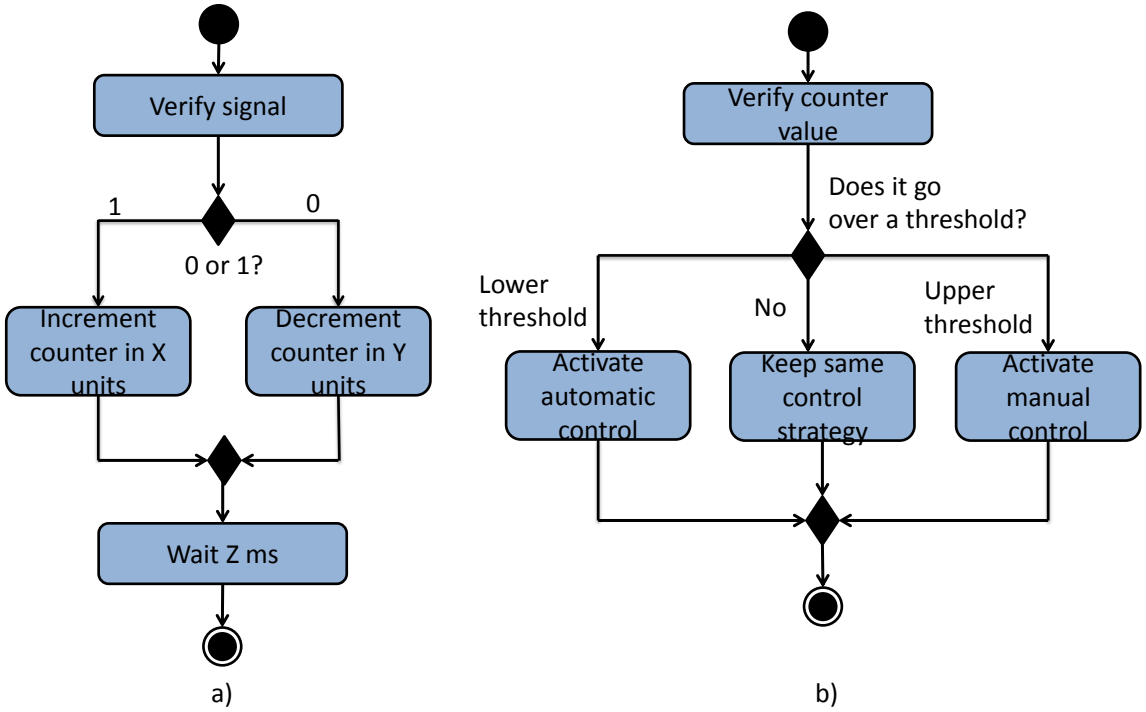


Figure 5.6: a)Counter implementation flowchart; b)Transition strategy flowchart;

Limiting the counter's upper and lower values (between 0 and 512) opens the possibility of stabilization. This way, and testing different values for X,Y and Z table 5.1 is obtained. Since the data obtained when the controller is off and on differs, thresholds can be established to signalize the right moment for changing the control strategy (5.6 b). To prevent unwanted rapid switching between strategies more than one threshold is established, making this process acquire an hysteresis effect.

Table 5.1: Test results for acquiring efficient threshold values.

Parameters			Maximum counter value	
Sampling Period (Z)	Increment (X)	Decrement (Y)	Controller: off	Controller: on
0.2 ms	10	1	272	70
	20	1	366	531
	30	2	540	540
0.33 ms	30	1	387	531
	40	2	532	156
	60	3	550	560
0.5 ms	20	1	313	541
	30	2	430	550
	50	2	558	570
1 ms	30	1	120	541

The best results occur with bigger sampling periods, which is advantageous because there is less processing load on the Arduino board. Initially the values chosen were $0,5ms$ for sampling period, $X = 30$ and $Y = 1$. This way it was possible to obtain a big gap in the maximum counter values for on and off. Later it was necessary to increase the sampling period to $1ms$ in order to not create conflicts with older algorithms, created in a previous work [Gil13]. For the same X and Y the thresholds for on and off maximum counter values kept separated enough to use this new solution. The threshold values used in the dissertation were 200 for the lower threshold and 300 for the upper threshold. Figure 5.7 shows the setup used by the hardware for this method to work.

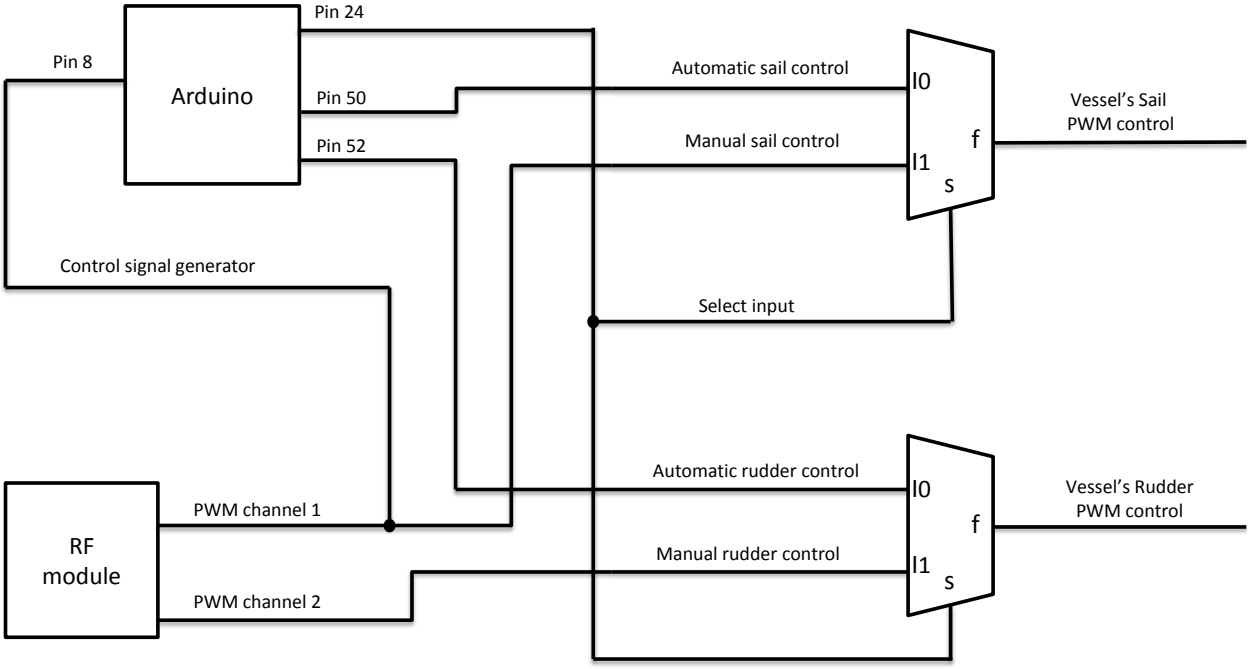


Figure 5.7: Pin scheme used for transition between controls.

5.2 Designed controller

Before designing a controller for the rudder and sail servomotors it is necessary to learn which variables are available from the selected sensor array. It is also important to know which of these variables will be useful for the controller's design.

The GPS sensor allows the attainment of the vessel's current position and speed and from the compass, current heading and inclination are obtained. The wind vane and anemometer sensors give the direction and speed of the wind, respectively. From these values alone it is not easy or direct to build a controller intuitively, for neither sail nor rudder. A new set of variables are needed to produce an easily understandable design, which is one of the objectives for this dissertation.

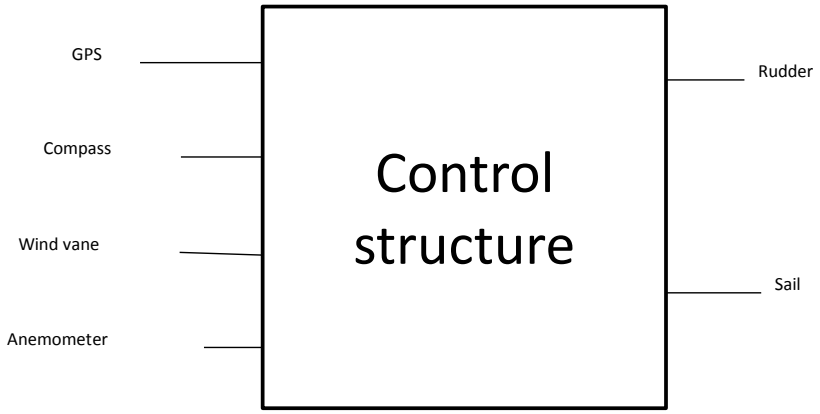


Figure 5.8: General layout of inputs and outputs of the controller system.

The variables shown in figure 5.9 were chosen for the first design of the fuzzy controller. Rudder and sail servomotor control are separated to reduce global complexity and also because there are situations, such as maintaining a course, where only one of the controllers needs to act. Some of the variables present are directly obtained from the sensors. Others such as wind and directional alignment need to be indirectly calculated.

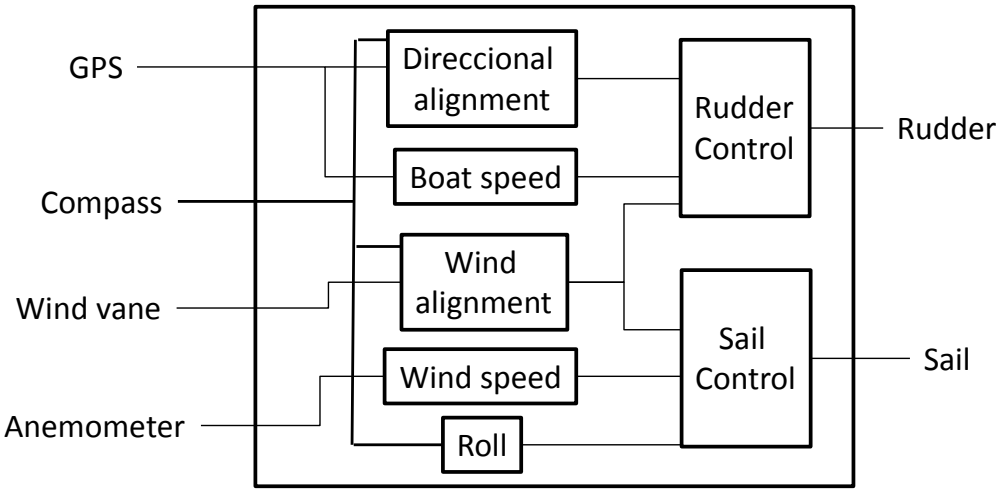


Figure 5.9: New set of variables used for sail and rudder control.

5.3 Controller Variables

5.3.1 Wind Alignment

The angle between the vessel's actual heading, given by the compass, and the wind direction, given by the wind vane, will be the new variable Wind Alignment. This variable's job is to tell in which direction the wind comes, from the vessels perspective. In sailing, the vessel can be divided in three zones; bow, lateral and stern with bow being the front, stern the rear and lateral the sides. Due to the sailboat's shape, these zones are mirrored and to simplify the variable, it is calculated to obtain values according to figure 5.10.

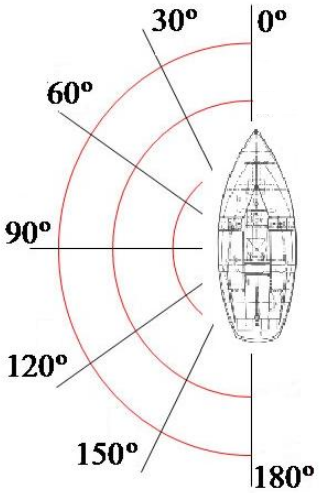


Figure 5.10: Wind alignment value range.

Its calculus consists in equation 5.1:

$$W.A. = |WindDirection - ActualHeading| \tag{5.1}$$

It is still possible to obtain angles in the $0^0 - 360^0$ range. To obtain angles in the $0^0 - 180^0$ range, algorithm 1 is implemented to the result of equation 5.1:

if $Wind\_Alignment > 180$ **then**
  $Wind\_Alignment = 360 - Wind\_Alignment$;
else
  Maintain current Wind_Alignment value;
end

Algorithm 1: Wind Alignment attainment

The examples in figure 5.11 show this reasoning:

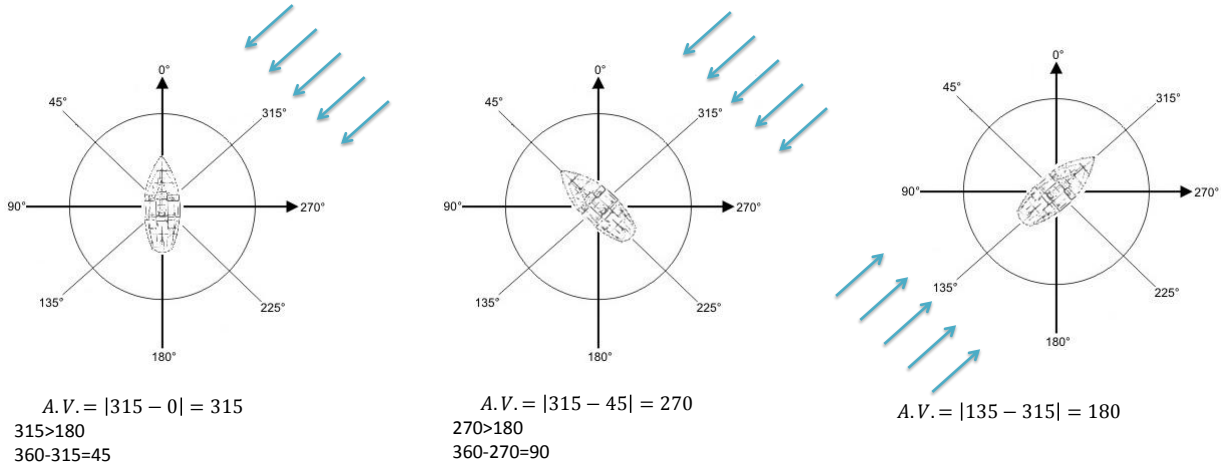


Figure 5.11: Wind alignment algorithm confirmation.

5.3.2 Directional Alignment

The angle between the vessel's actual heading and the intended course (the objective's position) will be the new variable Directional Alignment. Its job is to verify if the vessel is directionally aligned with the objective, as much as possible. This is needed to know which way the rudder must be turned to effectively follow the designated course.

To obtain this new variable, first it is necessary to obtain the actual heading, directly from the GPS sensor. The intended course is obtained using the vector between the actual position given by the GPS sensor and the objective coordinate. A simple graphical projection of the intended course is shown in figure 5.12.

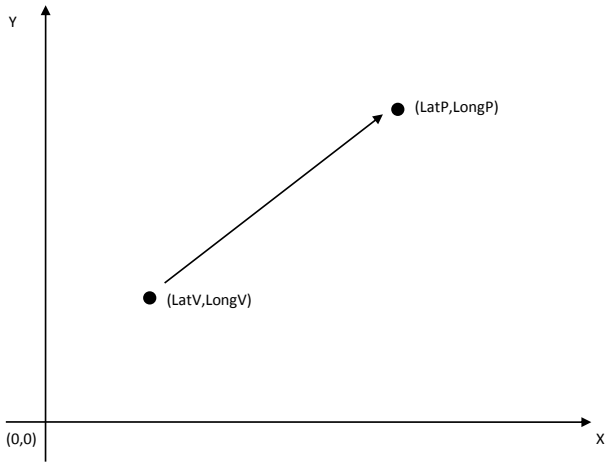


Figure 5.12: Original graphical projection of the intended course.

Since an angular value is intended, a conversion from rectangular to polar coordinates is needed. To simplify this conversion, the vector's point of origin (this is where the vessel is, (LatV,LongV)) should be changed to the axis origin (0,0). This is done by subtracting the value of origin (LatV,LongV) in both the coordinates, origin and destination, as shown in figure 5.13.

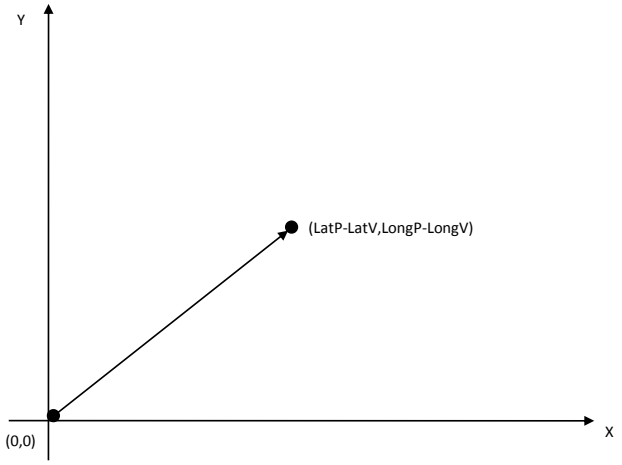


Figure 5.13: Modified graphical projection of the intended course.

With this projection and converting to polar coordinates, the angle between the intended course and the axis origin (representing the vessel) is obtained, as shown in figure 5.14.

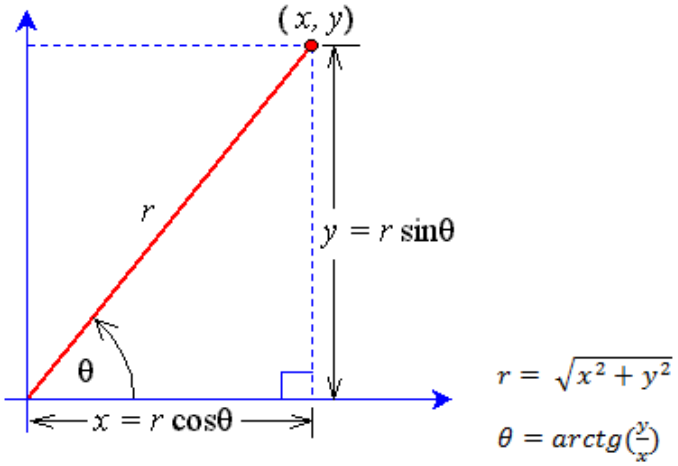


Figure 5.14: Rectangular to polar coordinate conversion, adapted from [Vau].

In order to obtain a referential to the vessel where each angle to its left is negative, and each angle to its right is positive, as shown in figure 5.15, more mathematical transformations are needed. The first problem is hardware related. The actual heading given by the compass sensor has its origin angle North (as such, it has 0^0 North on a compass chart) while the calculated intended course angle has its origin angle East (0^0 East by the same logic) because of the calculus method. As so, in every iteration of the directional alignment it is necessary to add 90 degrees to the intended course angle, to synchronize both angles. Knowing this, Directional Alignment is obtained using equation 5.2:

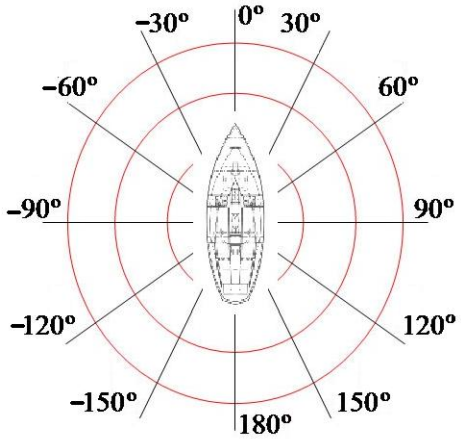


Figure 5.15: Directional alignment value range.

$$D.A. = ActualHeading - IntendedHeading \tag{5.2}$$

Also, since the work is done in the $[-180^0, 180^0]$ range instead of $[0^0, 360^0]$ it is necessary to verify if the value calculated is within the desired range, and if not, the algorithm 2 is applied:

if $Directional\_Alignment > 180$ **then**
  $Directional\_Alignment = Directional\_Alignment - 360$;
else if $Directional\_Alignment < -180$ **then**
  $Directional\_Alignment = Directional\_Alignment + 360$;
else
  Maintain current Directional_Alignment value;
end

Algorithm 2: Directional Alignment obtainment

The calculus done in table 5.2 confirm the effectiveness of this method:

Table 5.2: Confirmation examples for the directional alignment algorithm.

Intended c. at left of Actual c. Range: [-180,0]			Intended c. at right of Actual c. Range: [0,180]		
Actual-Intended	Result	Correction	Actual-Intended	Result	Correction
45 - 180	-135		0 - 270	-270	-270 + 360 = 90
180 - 300	-120		0 - 225	-225	-225 + 360 = 135
225 - 30	195	195 - 360 = -165	0 - 315	-315	-315 + 360 = 45
270 - 0	270	270 - 360 = -90	135 - 340	-205	-205 + 360 = 155
270 - 45	225	225 - 360 = -135	270 - 225	45	
315 - 115	200	200 - 360 = -160	315 - 180	135	

5.4 Control structure

The vessel has two different servomotors, one for adjusting the tightening of the sail sheet and another one for the rudder, used in steering the ship. The first division in the control structure will be the use of different controllers for different servomotors. On the sail part the structure is as simple as one controller. Because the dissertation is in a prototyping stage the main objective of this single sail controller is only to maximize the vessel's speed with strategies explained in sub-section 5.4.1 .

As for the rudder, the control structure is going to be more complex. Since there are several sailing strategies, depending on the sailing conditions there is an obvious need to use different controllers. The rudder structure created defines only three different controllers. These are for the situations of favorable wind conditions, and tacking (left or right). The obtained and final layout with the addition of these controllers is shown in figure 5.16.

Since all controllers have three variables, in order to display their structure in a more comprehensive and tabular way, one of the variable's value will be fixed. On the sail controller, the variable *Roll* is the fixed one, since it only has two values. On the rudder controllers the variable *Boat speed* has a fixed value in each rule table.

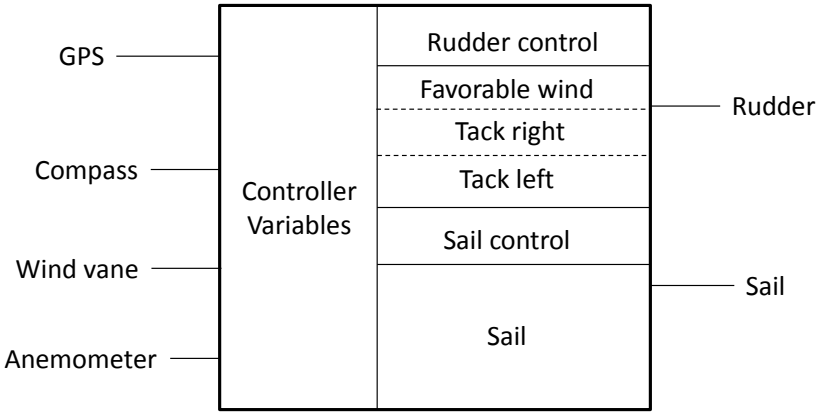


Figure 5.16: Updated layout of the controller system.

5.4.1 Sail

Before presenting the tables of rules and logic behind them, it is important to comprehend the fuzzy sets and membership functions of each variable. The fuzzy variables used in this controller are roll for inclination, wind alignment and wind speed. For all variables, the type of membership function used is the trapezoidal type (figure 3.9(b)). This choice relied mainly on simplicity of use for prototyping reasons, and because singleton type membership functions caused malfunctions on Xfuzzy simulations. Specification of each function required the use of four points, A, B, C and D. The first point is the one where the function starts to rise, the second where it stabilizes, the third is when it starts to fall and the last one is the point where the function reaches zero, as shown in figure 5.17. The value of each point for each membership function is shown in table 5.3.

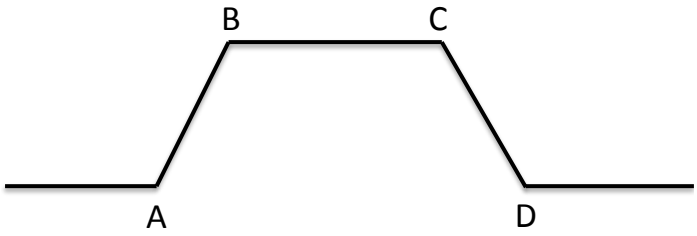


Figure 5.17: Trapezoid point specification.

Roll's fuzzy set consists of only two membership functions, for low inclination and high inclination. The controller only needs to know if the vessel is in risk of capsizing. The minimum inclination is 0^0 and the maximum is 90^0, no more is needed since a vessel inclined by 90 degrees has already capsized. The general look of this fuzzy set is shown in figure 5.18.

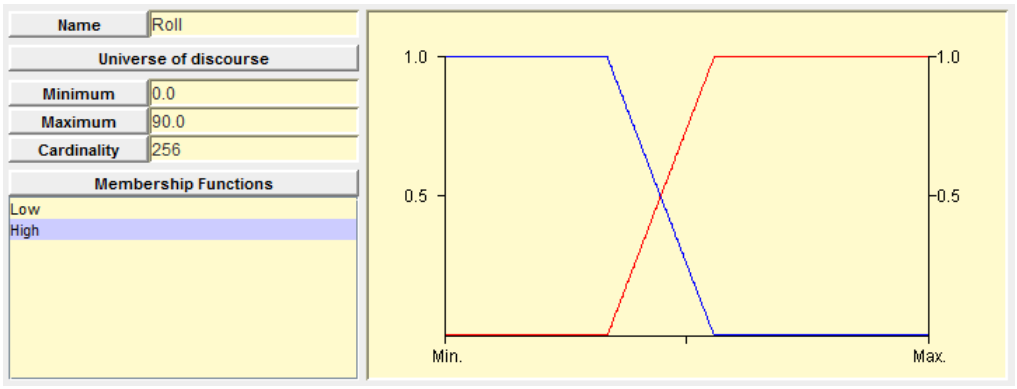


Figure 5.18: Roll's fuzzy set.

The windspeed's fuzzy set also consists on two membership functions, for slow or fast wind speed. If the wind is too rough, then the sails should be tightened in order to prevent capsizing or oversteering. The minimum value for this set is 0 knots and the maximum is 30 knots. These values are arbitrary, only physical testing can give the usable range. The general look of this fuzzy set is shown in figure 5.19.

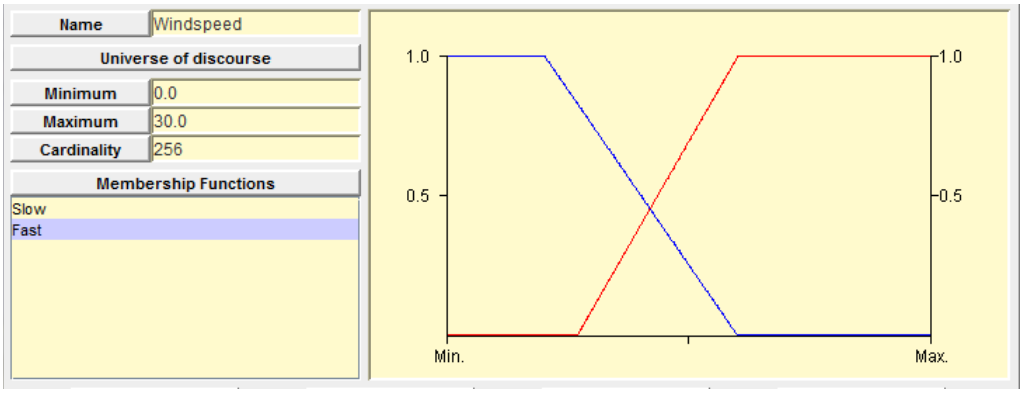


Figure 5.19: Wind Speed's fuzzy set.

Wind alignment has three functions. Each function represents a different zone of wind, bow being the front zone, lateral zone and stern being the rear zone of the vessel. This is the simplest way to describe the most important zones of wind. The minimum value is 0^0 and the maximum is 180^0, as explained in section 5.3.1. Figure 5.20 shows the general outline of this set.

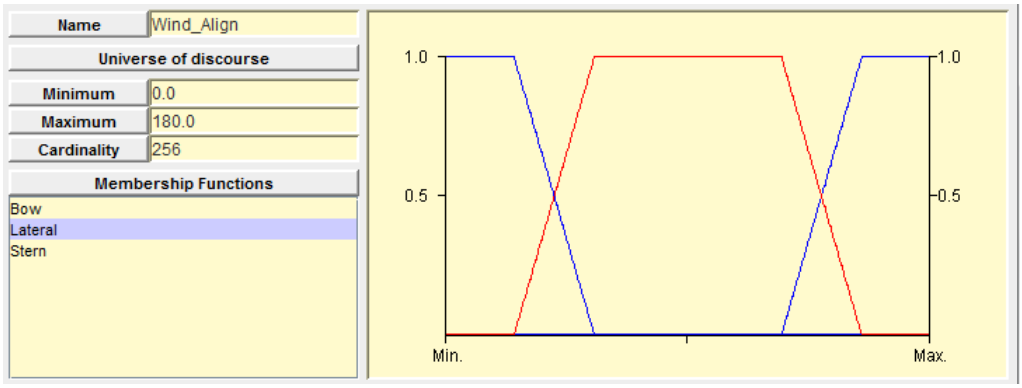


Figure 5.20: Wind Alignment's fuzzy set.

The sail fuzzy set, representing the output of the sail controller has five membership functions. These functions represent five different positions of the sail during navigation, two fastened, two loosened and one centered. The number five was chosen for simplicity of this proof of concept. Minimum and maximum values were arbitrarily chosen and can be later modified according to the output needed to move the vessel's servomotors. The trapezoid functions resemble singletons because the fuzzy inference used for these controllers is Sugeno type. Figure 5.21 shows the outline of this set.

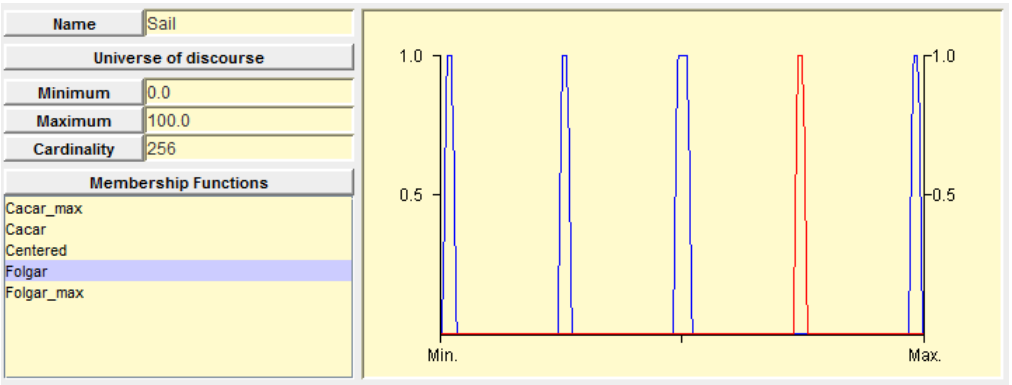


Figure 5.21: Sail's fuzzy set.

As mentioned before, the sail controller focuses on obtaining maximum speed efficiency. This is done by loosing or tightening the sail sheet according to certain conditions. For instance, if the vessel is facing downwind (with the wind on its back) the sail sheet should be as loose as possible to maximize the area of effect. In contrast, if the vessel is sailing upwind the sheet should be tightened to reduce wind resistance. It is also important to state that the sail controls most of the ship's inclination. If a ship is leaning too much, due to strong lateral winds, it can capsize and that is an obvious and important situation to avoid. As so, the sail sheet should be loosened as the ship is leaning. This enables most of the wind to pass through the sail and thus the vessel regains its natural balance. Table 5.4 shows the general outline of the sail inference system.

Table 5.3: Trapezoid values for each fuzzy set on the sail controller.

Sail		a	b	c	d
Roll	Low	-18	0	30	50
	High	30	50	90	108
Wind speed	Slow	-6	0	6	18
	Fast	8	18	30	36
Wind alignment	Bow	-16	0	25	55
	Lateral	25	55	135	155
	Stern	125	155	180	181
Sail	Tighten_Max	0	1	2	3
	Tighten	24	25	26	27
	Centered	48	49	50	51
	Loose	73	74	75	76
	Loose_Max	97	98	99	100

Table 5.4: Fuzzy rules for the sail inference system.

Roll:high

	Windspeed	
Wind Alignment	Slow	Fast
X	Loose	Loose_MAX

Roll: low

	Windspeed	
Wind Alignment	Slow	Fast
Bow	Tighten	Tighten_MAX
Lateral	Centered	Centered
Stern	Loose_MAX	Loose

5.4.2 Rudder

The rudder control is structured in order to steer the vessel, according to some conditions. It must avoid the upwind zone while steering to the objective. If it is needed to pass through, or if the objective is in the upwind zone, then tacking maneuvers must be deployed (as explained in subsection 3.1.3). As a result, the rudder control structure uses three different controllers, for tacking situations and normal maneuvers. These controllers are dynamically switched between them as the sailing progresses, in order to maintain a good course of action. The variables used in all three are the vessel's speed, wind alignment and directional alignment. Their fuzzy sets will be analyzed and trapezoid values (similar to table 5.3) are shown in table 5.5.

The directional alignment fuzzy set consists on six membership functions. Three represent the port side of the vessel and the remaining three represent starboard. The number of membership functions was chosen arbitrarily, and choosing a larger number of membership functions could mean less efficiency from the fuzzy controller. The minimum value is -180^0 and the maximum is 180^0, as explained in section 5.3.2. Figure 5.22 shows the general outline of this set.

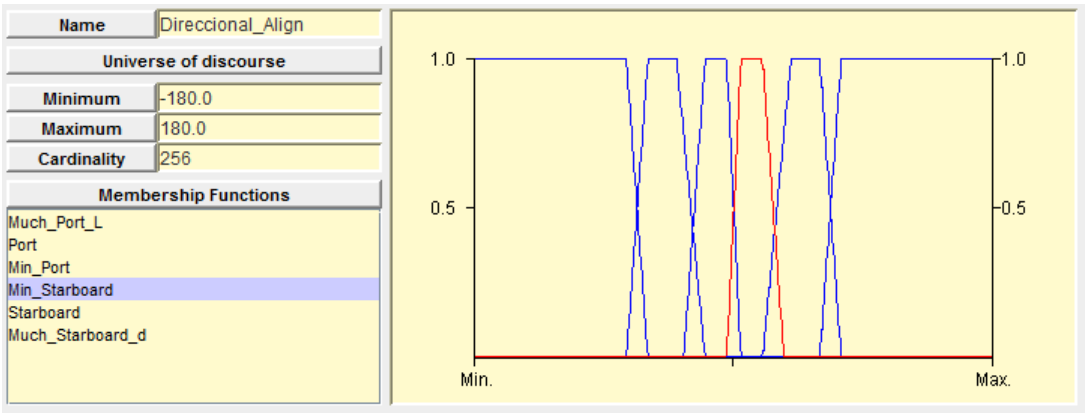


Figure 5.22: Directional Alignment's fuzzy set.

The vessel's speed fuzzy set (Boatspeed) consists on only two fuzzy sets, slow and fast. If the vessel is slow then turning should be more gentle, in order to maximize the gain in velocity. If it is fast then turning can be more rough. The minimum and maximum values are arbitrary in the same fashion as the sail's wind speed set. Figure 5.23 shows this set.

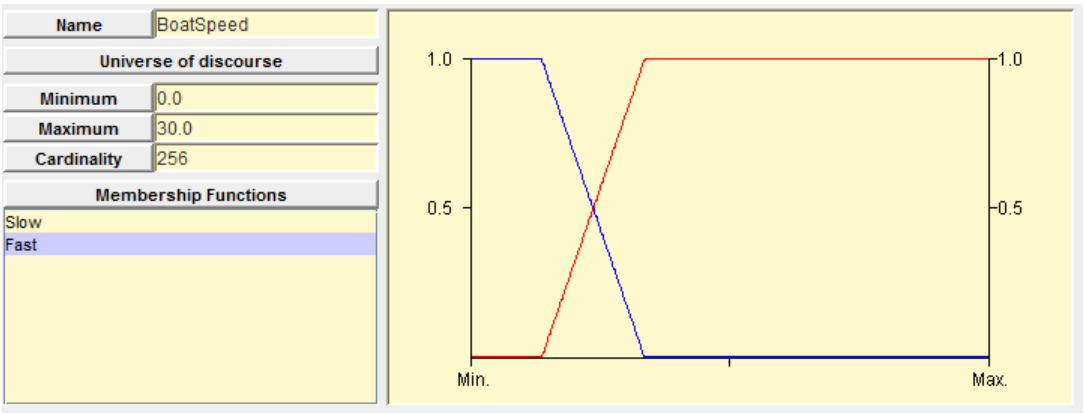


Figure 5.23: Boat Speed's fuzzy set.

The fuzzy set wind alignment is the same as the one described in the sail controller, thus it will not be analyzed again.

Finally, the rudder output fuzzy set consists in seven different membership functions. Three for each side (port or starboard) with different sensitivities and one for the center position. The logic for the number is the same as in the directional alignment and sail fuzzy sets. Minimum and maximum values were obtained during laboratory work

[SA13], using the simulator developed in [Gil13]. Also, the functions resemble singletons for the same reason explained in the Sail output fuzzy set. The general look of this fuzzy set is shown in figure 5.24.

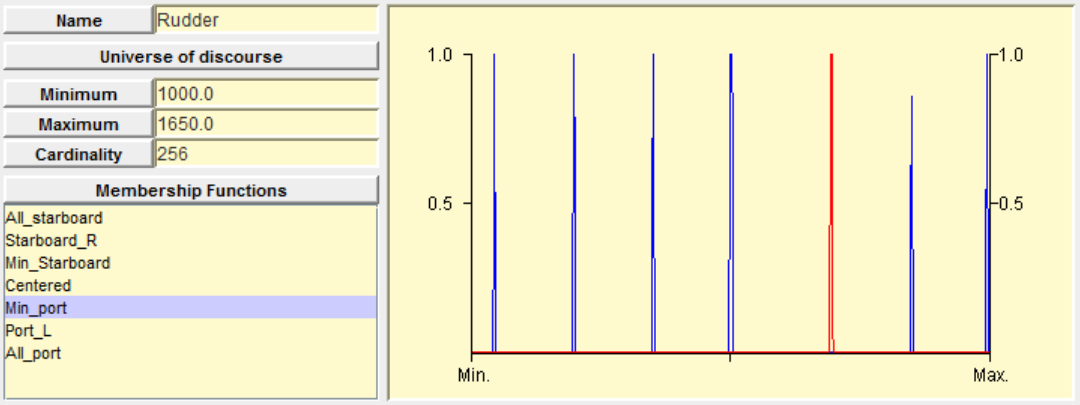


Figure 5.24: Rudder's fuzzy set.

Table 5.5: Trapezoid values for each fuzzy set on the rudder controllers.

Rudder		a	b	c	d
Direccional alignment	Much_port	-205	-180	-75	-60
	Port	-75	-60	-40	-20
	Min_Port	-35	-20	-5	5
	Min_Starboard	-5	5	20	35
	Starboard	20	40	60	75
	Much_Starboard	60	75	180	205
Boatspeed	Slow	-6	0	4	10
	Fast	4	10	30	36
Wind alignment	Bow	-16	0	25	55
	Lateral	25	55	125	155
	Stern	125	155	180	181
Rudder	All_starboard	1025	1026	1027	1028
	Starboard	1125	1125	1126	1127
	Min_Starboard	1225	1226	1227	1228
	Centered	1323	1324	1326	1327
	Min_Port	1450	1451	1452	1453
	Port	1551	1552	1553	1554
	All_Port	1646	1647	1648	1649

Favorable Wind

This controller simply sails directly to the objective. Because of this course of action, it should never enter an upwind zone (wind on front of the vessel) for it would stop moving. If this condition happens, the vessel steers in the opposite direction of the objective and is easy to deduce that staying away from the upwind zone is more important than reaching the objective in this mode. Table 5.6 shows the fuzzy rule set of this controller.

Table 5.6: Fuzzy rules for the rudder inference system - favorable wind situation.
Boat speed: Slow

	Wind alignment		
Directional Alignment	Stern	Lateral	Bow
Much_port	All_port	Port	Starboard
Port	Port	Min_port	Starboard
Min_port	Centered	Centered	Starboard
Min_starboard	Centered	Centered	Port
Starboard	Starboard	Min_starboard	Port
Much_starboard	All_starboard	Starboard	Port

Boat speed: Fast

	Wind alignment		
Directional Alignment	Stern	Lateral	Bow
Much_port	Port	Min_port	Min_starboard
Port	Min_port	Min_port	Min_starboard
Min_port	Centered	Centered	Min_starboard
Min_starboard	Centered	Centered	Min_port
Starboard	Min_starboard	Min_starboard	Min_port
Much_starboard	Starboard	Min_starboard	Min_port

Tacking left

When a tacking maneuver to the left is needed this is the chosen controller. The vessel steers completely to the left until it leaves the upwind zone. If the objective is still at left of the vessel's direction, it continues to turn left. If the objective is to the right after the tacking maneuver it means it is in an upwind zone. If this is the case the vessel is kept at bay from that zone until it is ready to make another tacking maneuver (using another controller for this effect). Table 5.7 shows the fuzzy rule set of this controller.

Table 5.7: Fuzzy rules for the rudder inference system - left tack situation.
Boat speed: Slow

	Wind alignment	
Directional Alignment	Lateral	Bow
Much_port	Port	All_port
Port	Port	All_port
Min_port	Port	All_port
Min_starboard	Centered	All_port
Starboard	Centered	All_port
Much_starboard	Centered	All_port

Boat speed: Fast

	Wind alignment	
Directional Alignment	Lateral	Bow
Much_port	Min_port	Port
Port	Min_port	Port
Min_port	Min_port	Port
Min_starboard	Centered	Port
Starboard	Centered	Port
Much_starboard	Centered	Port

Tacking right

This controller is very similar to the "Tacking left" controller described in section 5.4.2 and it only differs in direction. For instance, the vessel steers completely to the right leaving the upwind zone. If the objective is still to the right of the vessel after the tacking maneuver it continues to steer to the right. Table 5.8 shows the fuzzy rule set of this controller.

Table 5.8: Fuzzy rules for the rudder inference system - right tack situation.
Boat speed: Slow

	Wind alignment	
Directional Alignment	Lateral	Bow
Much_port	Centered	All_starboard
Port	Centered	All_starboard
Min_port	Centered	All_starboard
Min_starboard	Starboard	All_starboard
Starboard	Starboard	All_starboard
Much_starboard	Starboard	All_starboard

Boat speed: Fast

	Wind alignment	
Directional Alignment	Lateral	Bow
Much_port	Centered	Starboard
Port	Centered	Starboard
Min_port	Centered	Starboard
Min_starboard	Min_starboard	Starboard
Starboard	Min_starboard	Starboard
Much_starboard	Min_starboard	Starboard

5.4.3 Rudder decider structure

Because the rudder strategy is not always the same, it is necessary to dynamically change the controller according to the present environment. For this, a decider structure is implemented based on the available sensors and variables.

The first step is to decide which initial controller will be used and for this it is necessary to detect if the initial course is on an upwind course or not, using equation 5.3 and algorithm 3:

$$UpwindZone = |ActualHeading - WindDirection| \tag{5.3}$$

if $Upwind\_Zone > 315$ **then**
  $Upwind\_Zone = 360 - Upwind\_Zone$;
else
  Maintain current Upwind_Zone value;
end

Algorithm 3: Upwind_Zone obtainment

Then, if $U.Z. > 45$ the vessel is outside the upwind zone and the favorable wind controller is used. If not, it is still needed to know which tack controller is used, left or right. If the directional alignment is negative, the objective is to the left of the vessel and tack left controller is used. Is it is positive, tack right controller is used with the same logic.

After deciding the initial controller, the transitions between different controllers must respect some conditions. From direct navigation to tacking maneuvers the general condition is the entering in the upwind zone. This is detected when wind alignment is inferior to 45^o. Deciding which direction the vessel is going to tack is based on the value of the directional alignment. If positive, tack right is chosen and if negative tack left is chosen. From tacking to favorable navigation is simpler. This situation only happens if the objective is outside the upwind zone and in that case the vessel starts to gain distance from it. In this case, only the wind alignment needs to be taken into account. Finally, tacking left to tacking right and vice versa is decided using the vessel's speed and distance to the original trajectory, when the course was initiated. The general behavior of this structure is shown in figure 5.25.

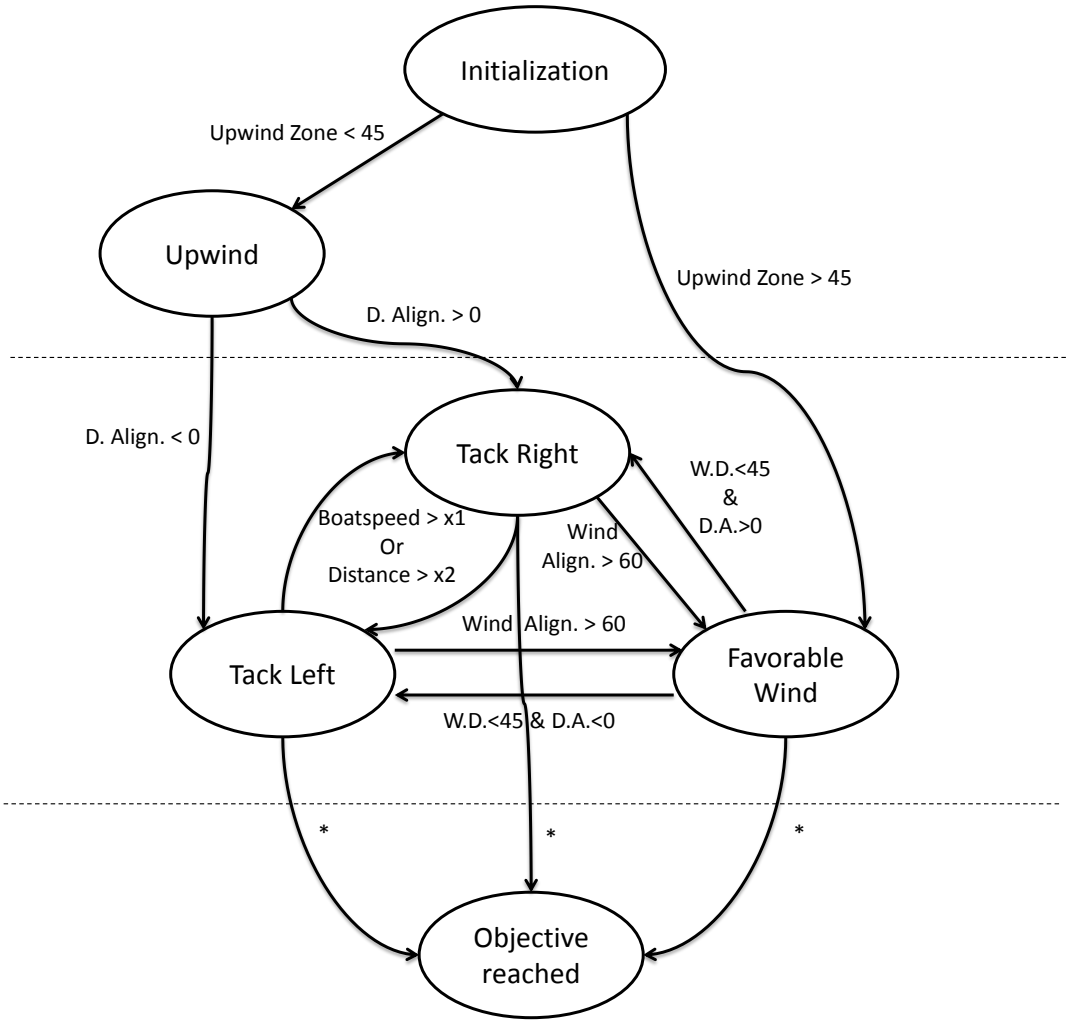


Figure 5.25: State machine of the rudder decider structure.

This length to the original trajectory is calculated using the notion of distance from a point to a line (d parameter in figure 5.26) [Dez13] and to find it, the notion of linear equation is used (equation 5.4).

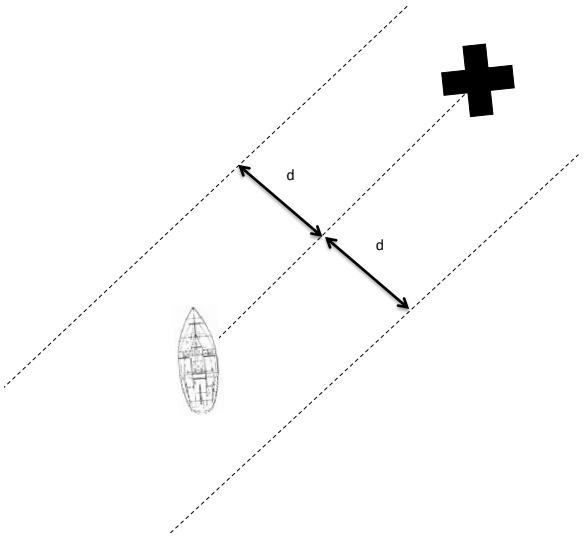


Figure 5.26: Potentially used sailing area, defined in the beginning of the course.

$$a.x + b.y + c = 0 \tag{5.4}$$

To determine equation 5.4, linear algebra concepts were used. Applying the rule of Sarrus [Kha10] to the determinant of a 3×3 matrix, the standard form of the linear equation is obtained. This is described in equation 5.5. Two pairs of coordinates $((x_1, y_1); (x_2, y_2))$ from all the possible ones where the line will pass are used.

$$Det \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x & y & 1 \end{vmatrix} = 0 \qquad \begin{aligned} a &= y_1 - y_2 \\ b &= x_2 - x_1 \\ c &= (x_1.y_2 - x_2.y_1) \end{aligned} \tag{5.5}$$

Using the initial position of the vessel and the point of origin as described in figure 5.12, equation 5.5 gives form to equation 5.6.

$$Det \begin{vmatrix} LatV & LongV & 1 \\ LatP & LongP & 1 \\ x & y & 1 \end{vmatrix} = 0 \qquad \begin{aligned} a &= LongV - LongP \\ b &= LatP - LatV \\ c &= (LatV.LongP - LatP.LongV) \end{aligned} \tag{5.6}$$

Finally, using equation 5.7 it is possible to obtain the distance to the original trajectory.

$$d = \frac{|a.x_0 + b.y_0 + c|}{\sqrt{a^2 + b^2}} \tag{5.7}$$

The values from both boat speed and distance need to be tuned according to the vessel used. This next stage is only possible if testing in real conditions, which was not done in the extent of this dissertation.

5.4.4 Goal approaching

Finally, while in the process of navigation, if the vessel is close enough to the objective it stops. This is done by aligning with the wind as explained in section 3.1.3. A simple controller for the rudder can put the vessel in the desired position as soon as it enters the goal range. Originally, there is no transition in modes for approaching the objective. It could be expected that while approaching the objective, the maximum distance for tacking decreases. All of this is an area open for development in another dissertation. There are two types of objectives targeted: buoys and finish lines.

Buoy approaching

A buoy is generally a floating object anchored to a specific location on the sea. These objects serve as the first target for using approaching techniques by the designed system. Since a commercial GPS module does not have the utmost precision (the one explained in section 4.2.3 has 10 meter positional accuracy) it is very difficult to know the exact location of the buoy. Establishing a radius around its expected position is a good way to know if the vessel is near this objective, and if so it can stop.

Figure 5.27 shows the way this approach was calculated. Since the distance from the vessel to the objective is known at every iteration of the course, it can be used as a second radius (r2). If this radius becomes smaller than the one established by the user (r1) it means that the vessel is close enough to its objective. This way, determining this approach is as simple as $r1 < r2$.

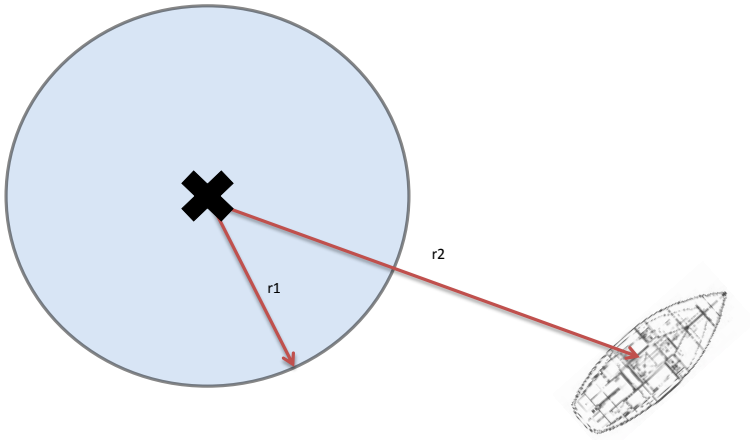


Figure 5.27: Radius from both the objective buoy and distance to the vessel.

Finish line approaching

In all kinds of races, the final objective is to pass the finish mark, preferentially in first place. In maritime races this is also an objective and so, the need to detect the passage through this line is essential in autonomous sailboats.

Given the coordinates of both end points of the finish line it is possible to obtain its linear equation following the method used in section 5.4.3, equation 5.5. Using the linear equation of the finish line in the form shown in equation 5.8 both half-planes formed by the finish line in a Cartesian plane are easily identified, as shown in figure 5.28 a).

$$y = -\frac{a.x + c}{b} \tag{5.8}$$

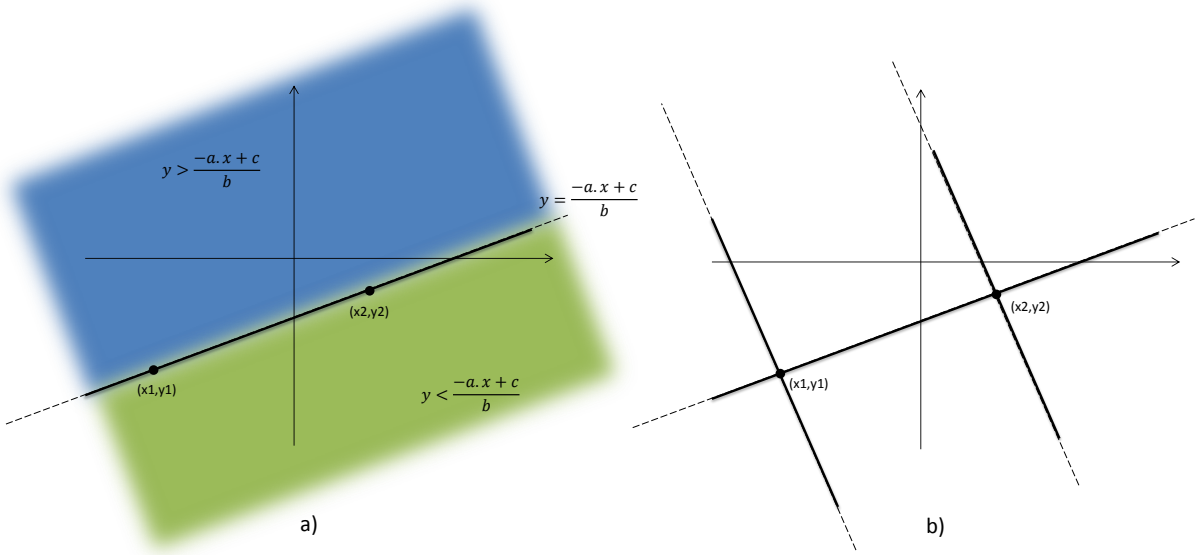


Figure 5.28: a) Finish line half planes identification; b) Lines at right angles to the finish line;

Also, it is important to know if the vessel has precisely passed through the finish line and not by its sides. For this, knowledge of the linear equations at right angles with both end points of the finish line is relevant. These lines are shown in figure 5.28 b). Being the slope of the finish line $m_{fn} = 1$, for an equation perpendicular to this one its slope is its negative ($m_{fn}.m_p = -1 \Leftrightarrow 1.m_p = -1 \Leftrightarrow m_p = 1$). Knowing this slope and the end point coordinate (X_a, Y_a) it is possible to obtain the perpendicular linear equation as shown in equation 5.9.

$$y - y_a = -1(x - x_a) \Leftrightarrow y + x - y_a - x_a = 0 \tag{5.9}$$

By means of numerical comparisons described in algorithm 4 it is possible to identify each of the zones in figure 5.29. These zone separations are essential for the implemented method to function. The main zones are the upper (5) and lower zone (4) of the finish line. Also, three more zones are used to identify if the vessel is between both end points of the finish line (2) or out of bounds (1 and 3). Applying algorithm 5 ensures finish line

detection.

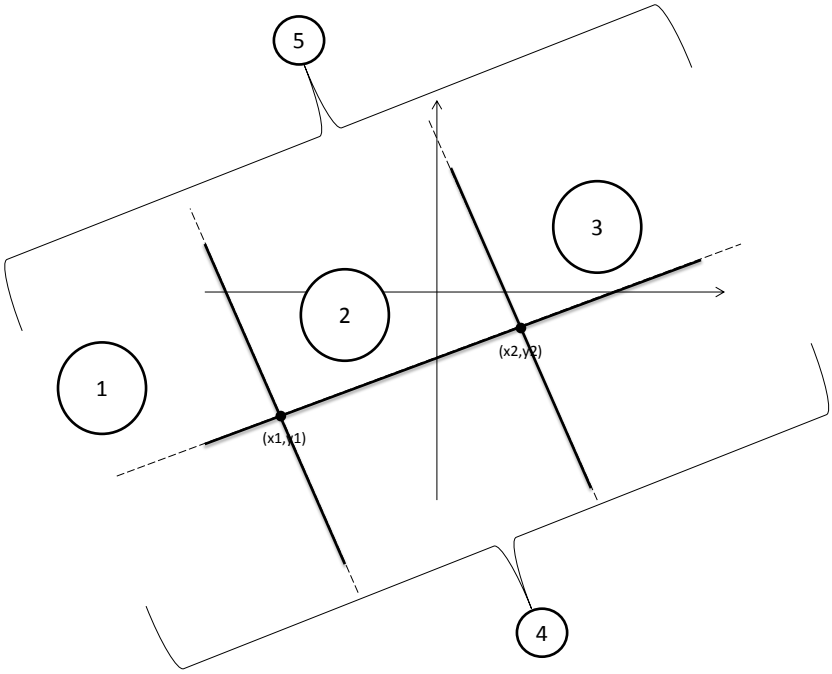


Figure 5.29: Zone division for finish line detection.

Using the finish line linear equation;
if $y > -\frac{a.x+c}{b}$ **then**
| The vessel is in zone 5;
else if $y < -\frac{a.x+c}{b}$ **then**
| The vessel is in zone 4;
end

Using the linear equations that pass through (x_1, y_1) and (x_2, y_2);
if $y_1 > -\frac{a_1.x_1+c_1}{b_1}$ *and* $y_2 < -\frac{a_2.x_2+c_2}{b_2}$ **then**
| The vessel is in zone 2;
else
| The vessel is out of zone 2;
end

Algorithm 4: Finish line zone detection algorithms

```
/* Initializations                                                    */
if y > -(a.x+c)/b then
|  Initial plane=5;
else
|  Initial plane=4;
end
Actual plane = Initial plane ;
/* At each iteration of the algorithm                                 */
if y > -(a.x+c)/b then
|  Actual plane = 5;
else if y < -(a.x+c)/b then
|  Actual plane = 4;
end
if Actual plane == Initial plane then
|  Finish line not yet reached;
else Finish line half plane crossed
|  if y1 > -(a1.x1+c1)/b1 and y2 < -(a2.x2+c2)/b2 then
|  |  Finish line crossed;
|  else
|  |  Course to finish line failed, reroute is needed;
|  end
end
```

Algorithm 5: Finish line crossing detection algorithm

Some final considerations and additional specifications on this method are presented next:

- The method described above, more specifically zone 2 detection in algorithm 4 produces results if the first coordinate (x_1, y_1) has a lower value in y than the second coordinate (x_2, y_2). If the opposite is the case, then both mathematical operators in the if condition are reversed ($y_1 < -\frac{a_1.x_1+c_1}{b_1}$ and $y_2 > -\frac{a_2.x_2+c_2}{b_2}$ instead of $y_1 > -\frac{a_1.x_1+c_1}{b_1}$ and $y_2 < -\frac{a_2.x_2+c_2}{b_2}$).

- Generally, the end points of a finish line are marked with buoys. To avoid crashing, when using finish lines it is important to mark the distance for tacking (d parameter in figure 5.26) smaller than the distance between both end points.

- Still concerning the last point, making a turn around a buoy can be seen as a succession of finish line crossings. If implementing these crossings for making a turn, the user should mark the objective point as near as possible to the buoy, with a minimal safe distance to avoid crashing.

6

Experimental results

In this chapter results from the XFuzzy libraries will be shown, via the Arduino serial monitor. To obtain data through the Arduino, code integration was done as explained in section 6.1. Graphical representations are shown in order to attain a better understanding. The data used in these representations is shown in appendix A and it was obtained within a laboratory work [SA13]. Different input sets were given to each controller in order to test the output range, and if this range is different whether on the Arduino or the Xfuzzy environment. In order to acquire this difference, the percentage error was calculated between the Xfuzzy and Arduino's outputs using equation 6.1.

$$\left| \frac{Output_{xfuzzy} - Output_{arduino}}{Output_{xfuzzy}} \times 100\% \right| \tag{6.1}$$

The controllers for the Favorable Wind, Tacking Left and Tacking Right situations have the same fuzzy variables, being these "Rboatspeed", "Rdir_align" (directional alignment) and "Rwind_align" (wind alignment). Sail control has two different fuzzy variables ("Sroll" (vessel's inclination) and "Swind_speed") and one common variable with the rudder, "Swind_align". For better understanding it is convenient to identify every variable value range. Rboatspeed and Swindspeed have arbitrary values ranging from 0 to 30 since the sensors for measuring them weren't available. Rdir_align varies from -180^0 to 180^o (as explained in section 5.3.2), Rwind_align/Swind_align from 0^o to 180^o (section 5.3.1) and Sroll varies from 0^o to 90^o, which are the only necessary angles for knowing if the ship capsized or not. Rudder output varies from 1025 and 1650 which are the values in milliseconds used by the rudder control system for starboard and port, respectively [Gil13]. Sail output varies from 0 and 100 which represent the sail tightened and loose, respectively.

6.1 Xfuzzy- Arduino integration

Xfuzzy's synthesized C++ descriptions are not 100% compatible with the arduino libraries. In order to integrate these generated libraries with the arduino code, the following changes need to be made in the libraries' code and in the arduino root.

1. Two new functions need to be added to the core Arduino functions on files new.cpp and new.h. These functions serve for memory allocation and freeing with arrays and are the following:

```
void * operator new[](size_t size) {
return malloc(size);
}

void operator delete[](void * ptr) {
if(ptr) free(ptr);
}
```

2. on the created .hpp file, file extension must be changed to .h instead and the names of "and", "or" and "not" operators also need to be altered to not enter in conflict with Arduino's reserved words.

3. on the created .cpp file, the name of the operators op.and, op.or and op.not must also be changed.

Automating this process is less time consuming on long term, as such it is an option to consider in terms of project improvement.

Also, another problem that occurred during development was memory overflow. Arduino is not equipped with a large memory bank and each iteration of the fuzzy inference engine consumed memory. Consequently, after all memory is used the program froze. This is probably related to faulty memory freeing functions and it is crucial that it is solved, in a later stage of the prototype development.

6.2 Experimental results and analysis

The results obtained from figures 6.1 to 6.18 used code generated from libraries obtained via Xfuzzy's synthesis tool, version $3.3b1$. Since results from the Arduino and XFuzzy are very similar, as shown in appendix A, the figures shown next only represent data obtained from Arduino. Each figure is analyzed after it is shown in order to gain a better understanding.

6.2.1 Sail

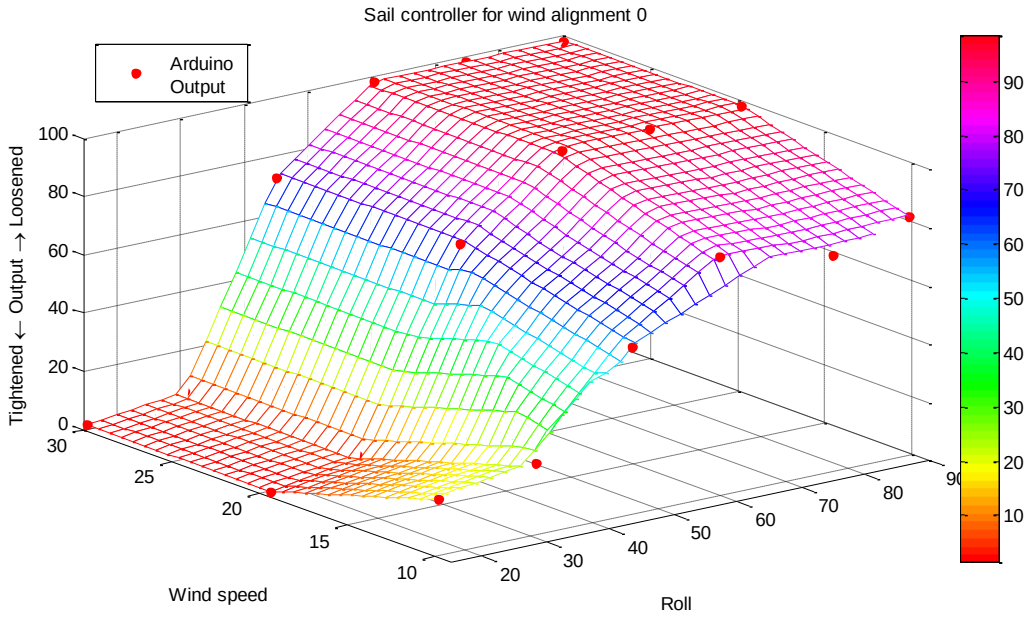


Figure 6.1: Experimental results for the sail controller part 1.

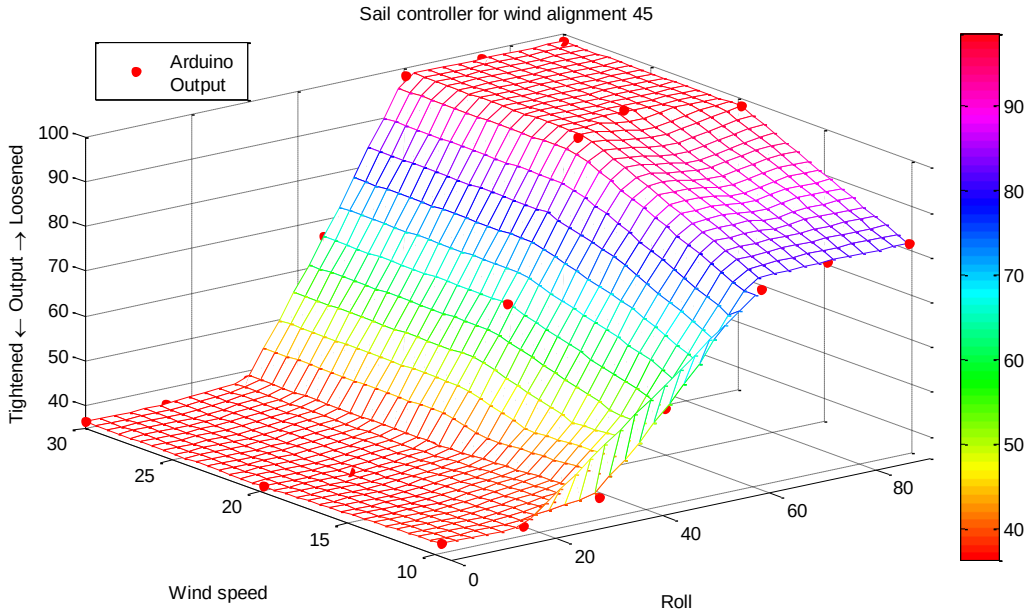


Figure 6.2: Experimental results for the sail controller part 2.

With the wind in front of the vessel (Wind alignment = 0) the sail tightens itself to the maximum in order to catch the less possible wind. In this figure and in every other sail controller result, if the inclination gets too steep then the sail loosens its sheets to the maximum, as explained in section 5.4.1. As the wind alignment rises, the vessel starts to catch the wind laterally. With wind alignment at 45 degrees the sailboat loosens its sheets a bit more. It is also important to notice that the sail control changes with the wind speed. If the wind is strong, then the sail is tightened a bit more than if the wind was weak.

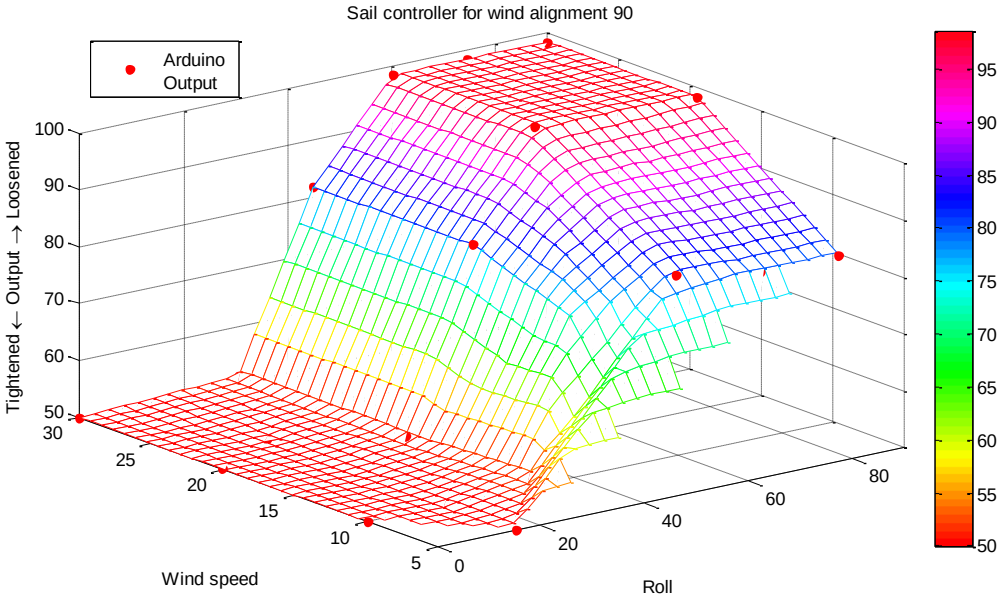


Figure 6.3: Experimental results for the sail controller part 3.

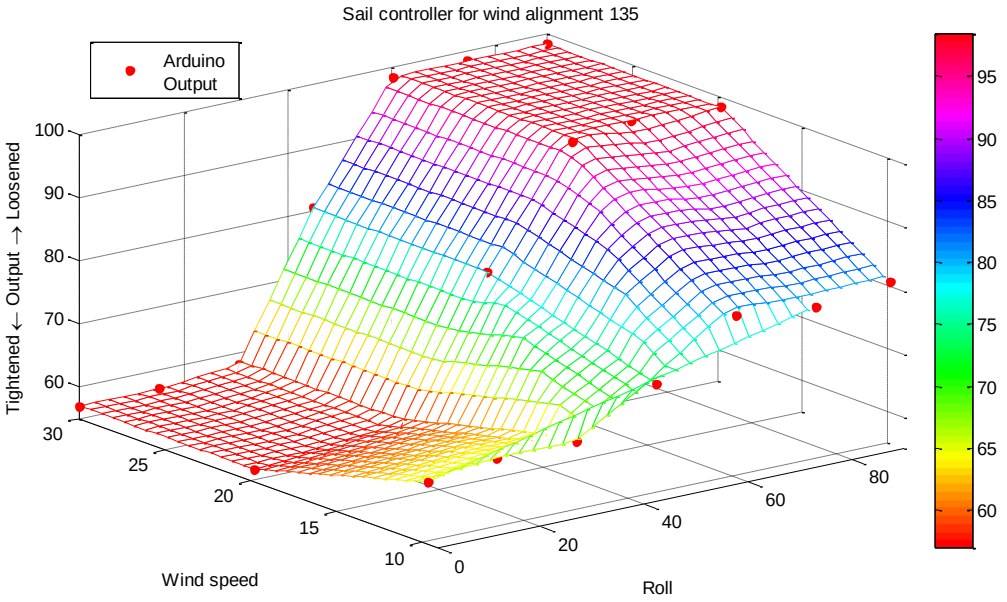


Figure 6.4: Experimental results for the sail controller part 4.

With the wind at right angles, the sails are tightened at their middle position, as it can be noticed the tendency is to loosen the sail the more the wind comes from behind. With 135 degrees of wind alignment the sail continues relatively in its middle position. As it can be noticed, the tendency for maximum loosing of the sails if the inclination is high continues with these values.

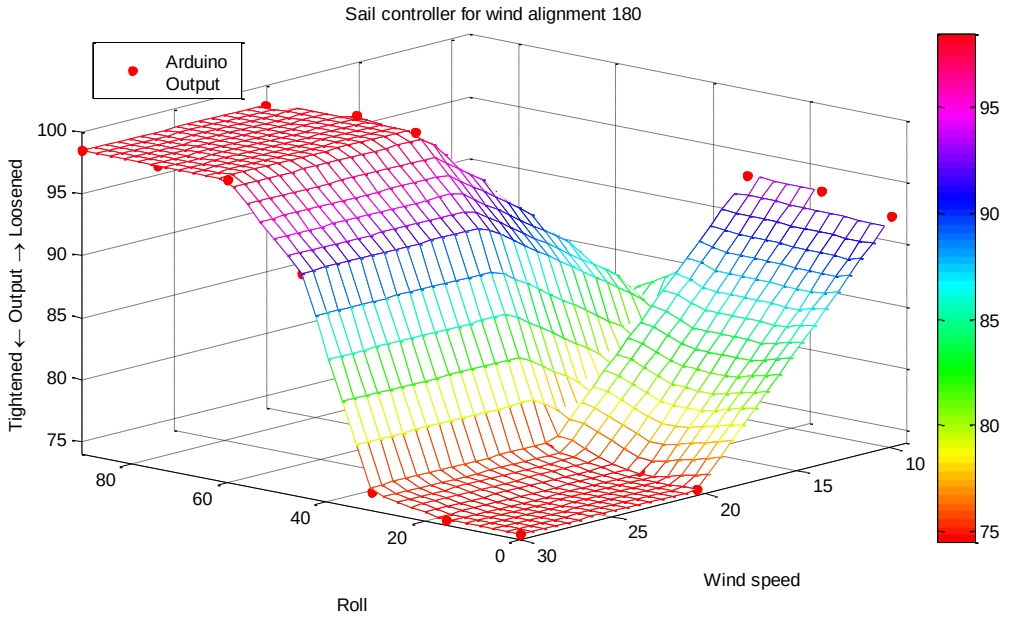


Figure 6.5: Experimental results for the sail controller part 5.

With the wind coming from behind the ship the sails are loosened in order to resist the most to the passage of the wind. Again, with a more steep inclination the controller focuses on loosing the sail to it's maximum to avoid capsizing. If the wind is weak then the sail is loosened. If it is strong then is it loosened to the maximum.

6.2.2 Favorable Wind

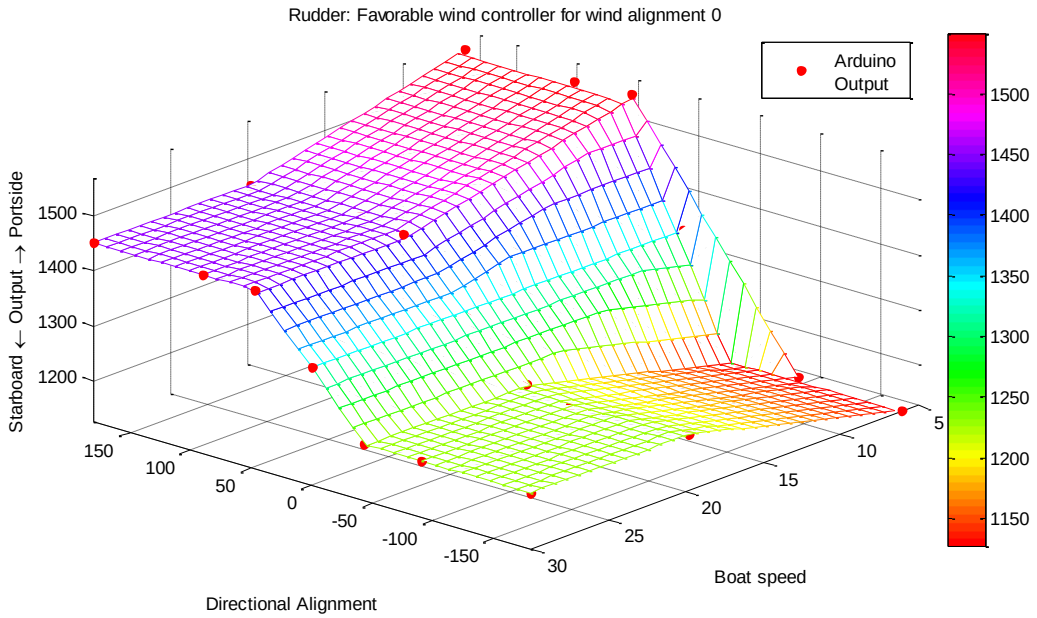


Figure 6.6: Experimental results for the favorable wind controller part 1.

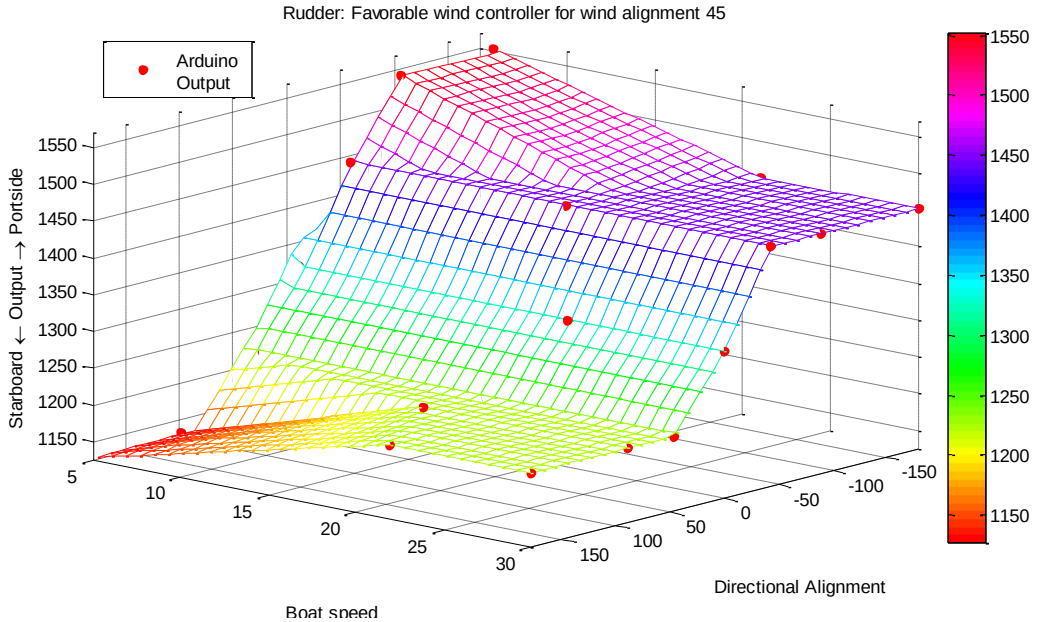


Figure 6.7: Experimental results for the favorable wind controller part 2.

In this controller, the situation of wind directly in front never happens. Even though, if some failure occurs and this controller is chosen in this situation it acts as appropriately as it can. If the directional alignment has 0 degrees in angle (the objective is directly ahead) then the controller cannot choose which course of action to take. If it is even a bit to the left (negative directional alignment) or to the right (positive directional alignment) then the rudder changes abruptly to turn the vessel away from this zone. In the rudder controllers the tendency to turn more softly if the vessel has a greater speed follows the same logic as the tightening of the sail at greater wind speeds. With wind alignment at 45 degrees the vessel starts to move more freely. If the objective is to the left (negative directional alignment) then the vessel simply sails to portside. If the objective is to the right it follows the same logic and sails to starboard. As the objective is more centered to the vessel's position the rudder starts to turn less and less, in order to make the most precise approach possible.

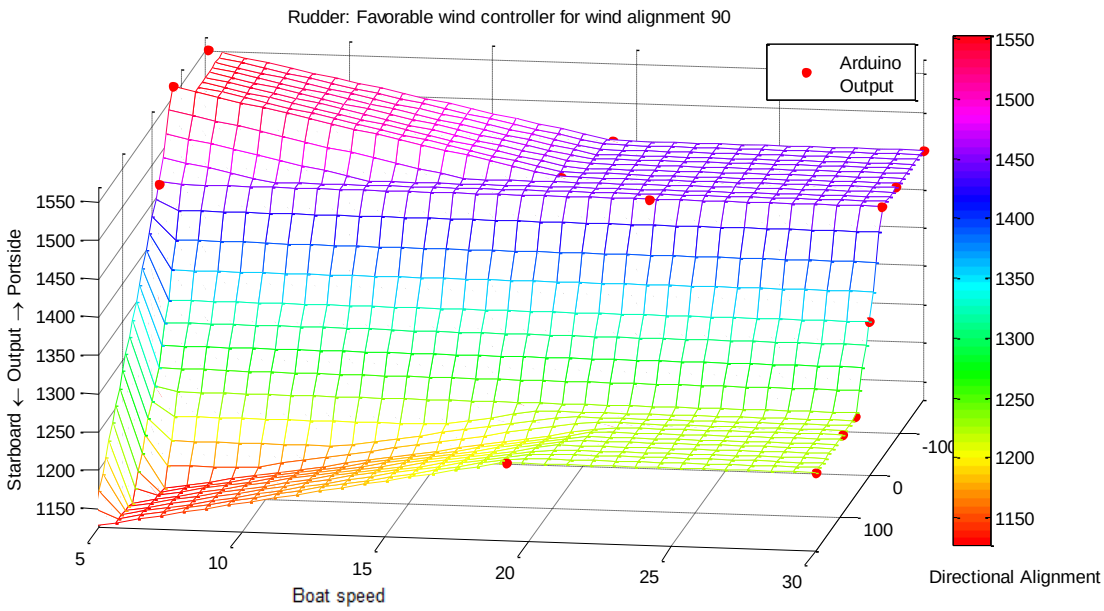


Figure 6.8: Experimental results for the favorable wind controller part 3.

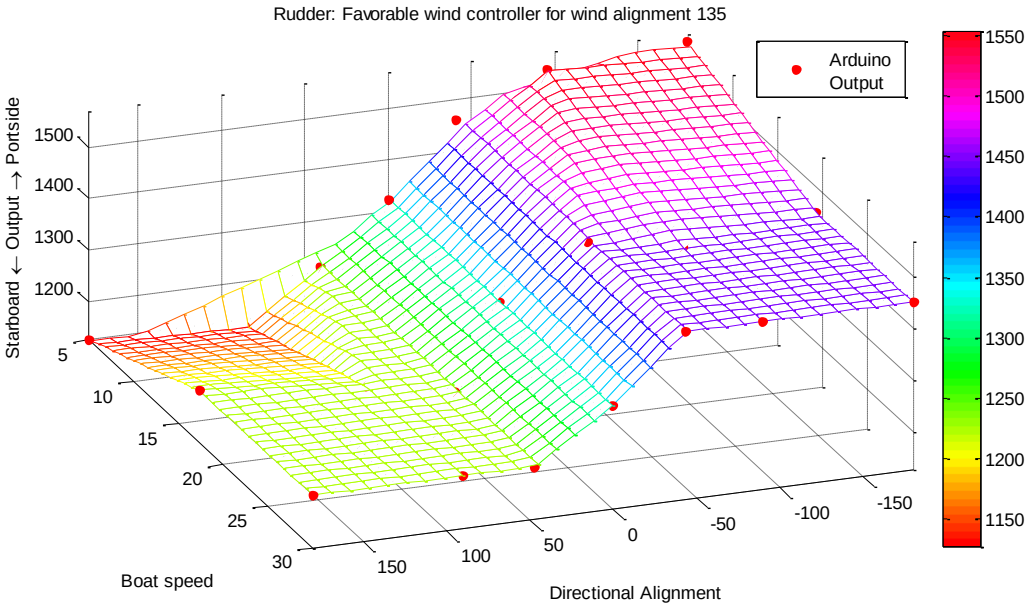


Figure 6.9: Experimental results for the favorable controller part 4.

With wind alignment from 90 to 180 degrees the behavior is similar to when wind alignment is 45 degrees. The only difference resides in the roughness of the rudder turn. If the objective is to the left then the vessel turns left, if the objective is to the right then the vessel turns in this direction. The tendency to turn more smoothly when the vessel is going "faster" continues. It is also important to notice that the situations where wind alignment is 90 and 135 degrees are equivalent because the controller marks both situations as lateral wind situations, described in subsection 5.4.2.

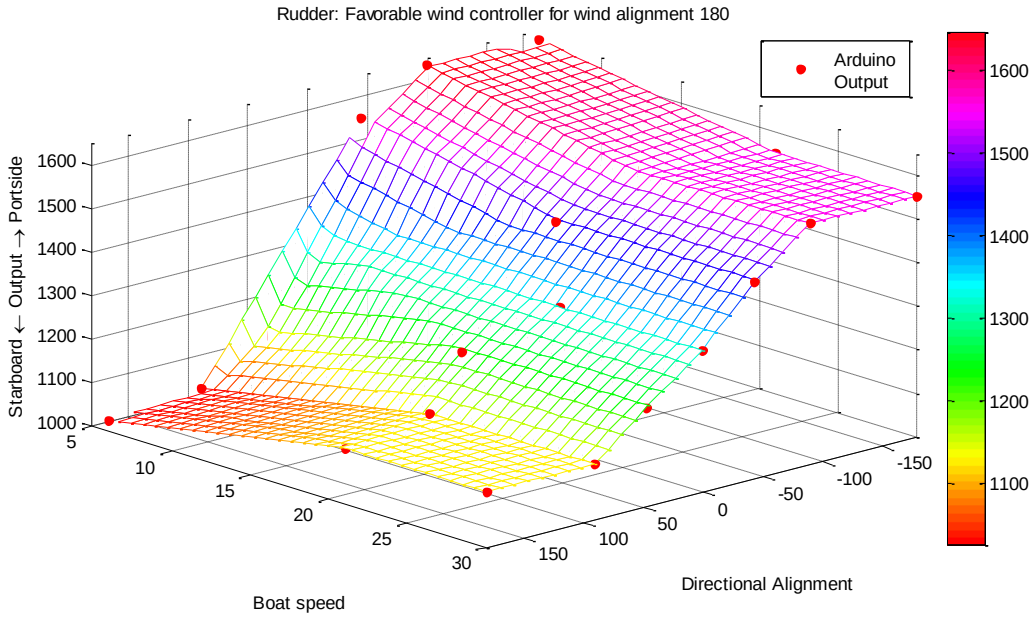


Figure 6.10: Experimental results for the favorable wind controller part 5.

6.2.3 Tack Left

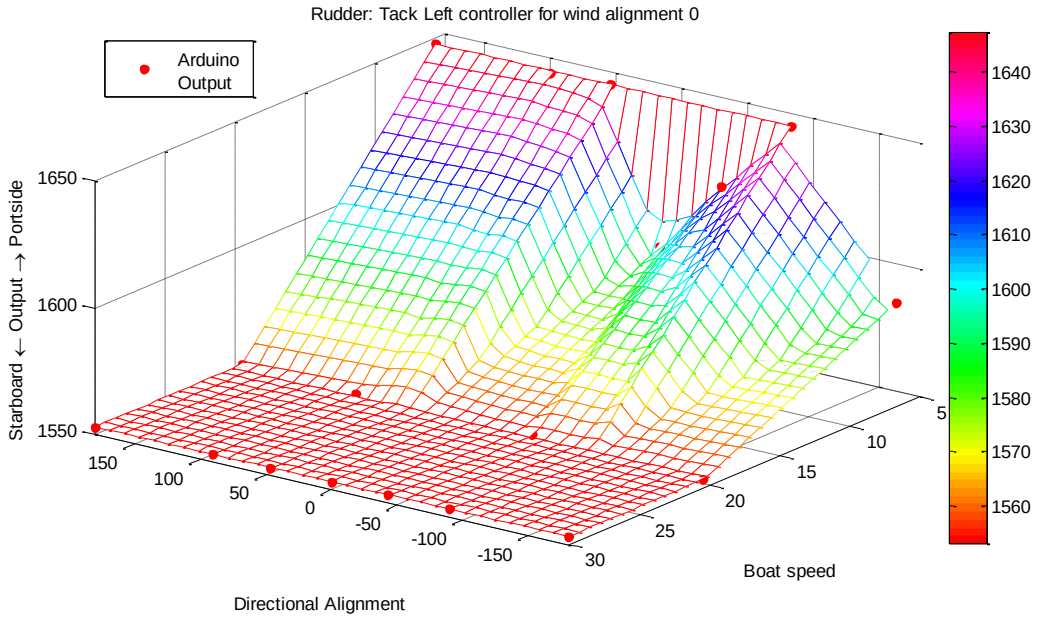


Figure 6.11: Experimental results for the tack left controller part 1.

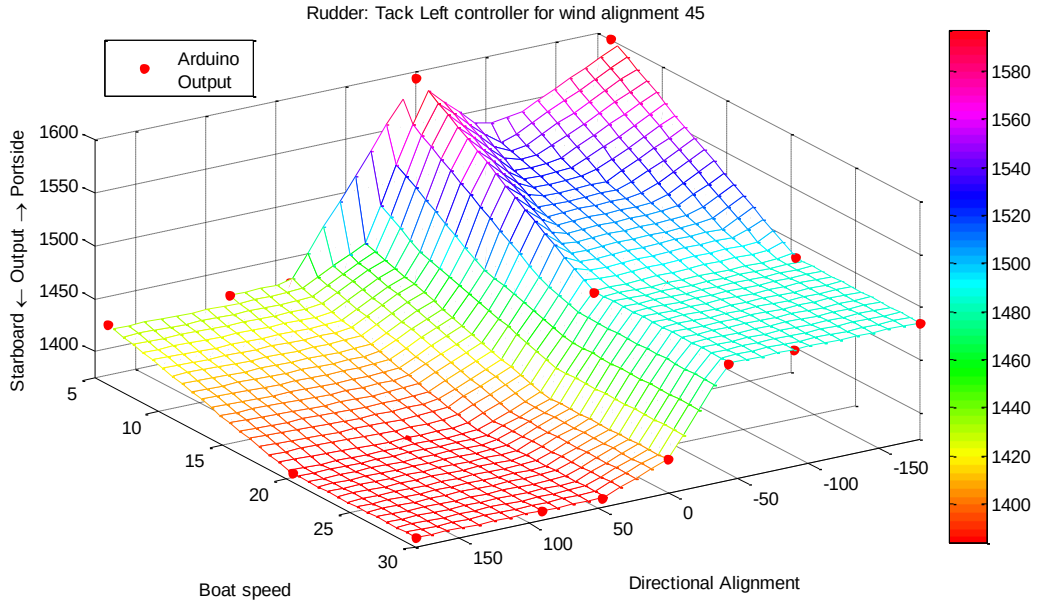


Figure 6.12: Experimental results for the tack left controller part 2.

The tack left controller's purpose is to pass the upwind zone if the vessel is to the right of the objective and must pass through it. With this specification, it is normal maneuvering if the vessel turns the rudder all to the left (portside) if the wind alignment is 0 degrees (wind directly in front).With wind alignment at 45 degrees it is safe to sail. Still, going with the specification, if the objective is to the left of the vessel, it continues turning in that direction. If the objective is to the right it is still in the upwind zone. In this case, the rudder is centered, keeping the vessel in the border between the sailing zone and

the upwind zone until it is ready to make another tacking maneuver. This new tacking maneuver is set by the rudder decider structure described in section 5.4.3.

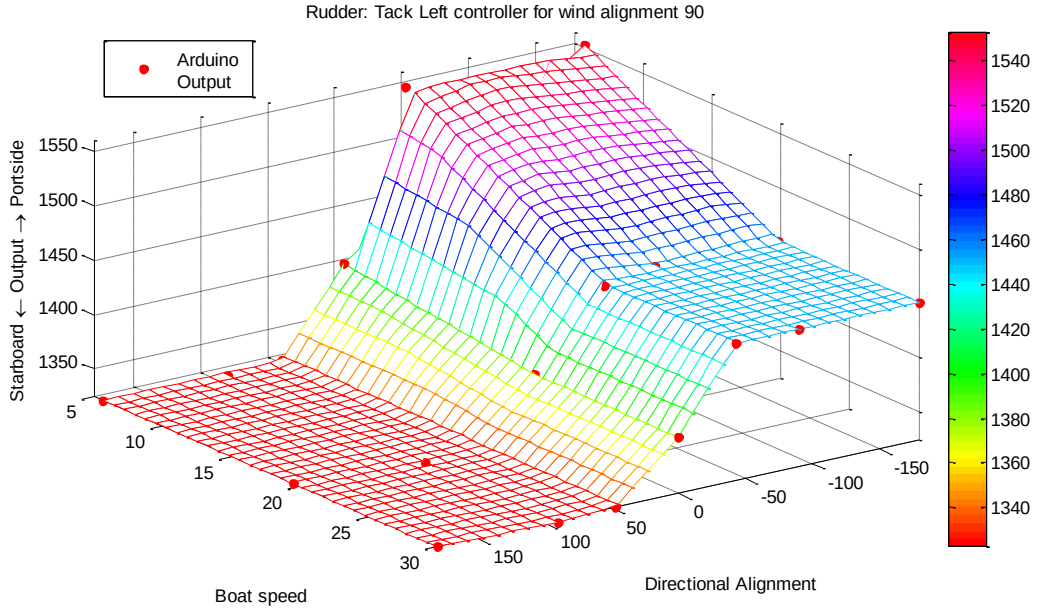


Figure 6.13: Experimental results for the tack left controller part 3.

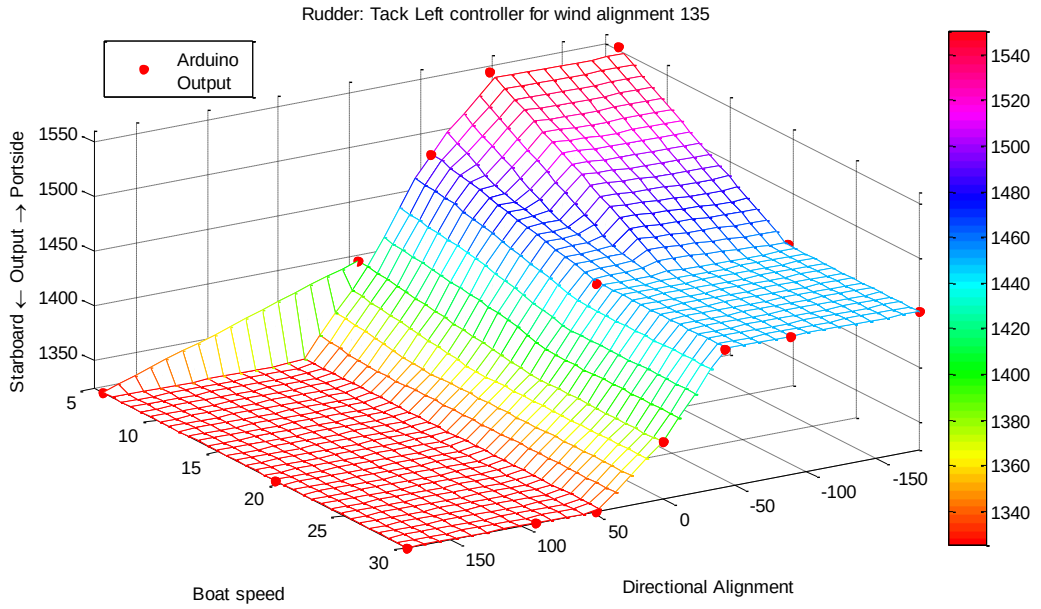


Figure 6.14: Experimental results for the tack left controller part 4.

With lateral wind (90 and 135 degrees) the strategy continues. If the objective is to the left of the vessel, then it continues turning left until the favorable wind controller is switched on. If the objective is to the right of the vessel then it is still in the upwind zone and the vessel keeps at bay from it. The tendency to turn more smoothly when the speed is greater continues in all tack left results.

6.2.4 Tack Right

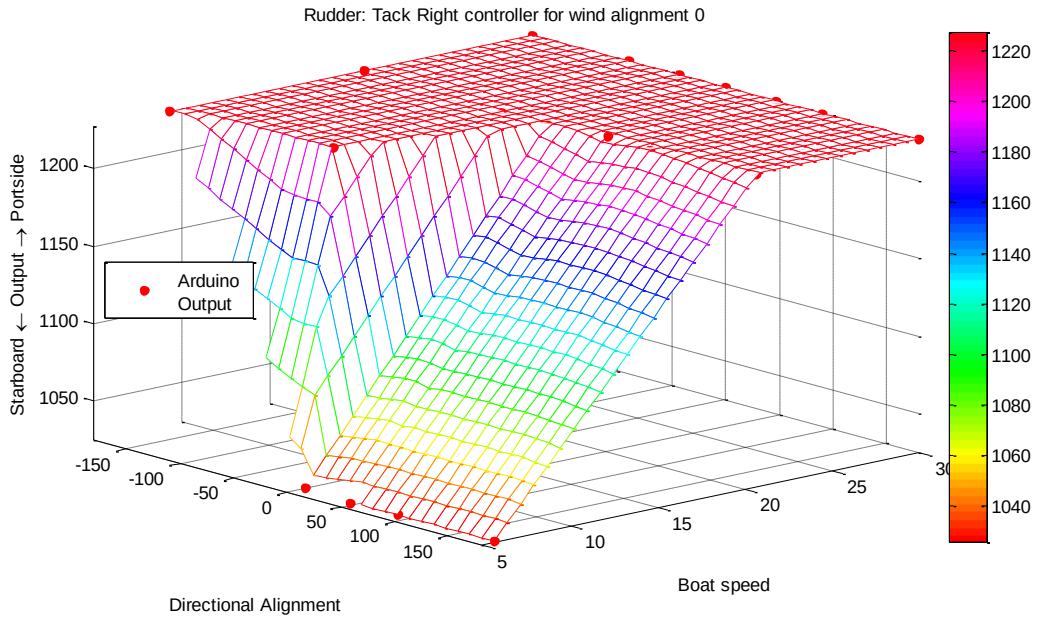


Figure 6.15: Experimental results for the tack right controller part 1.

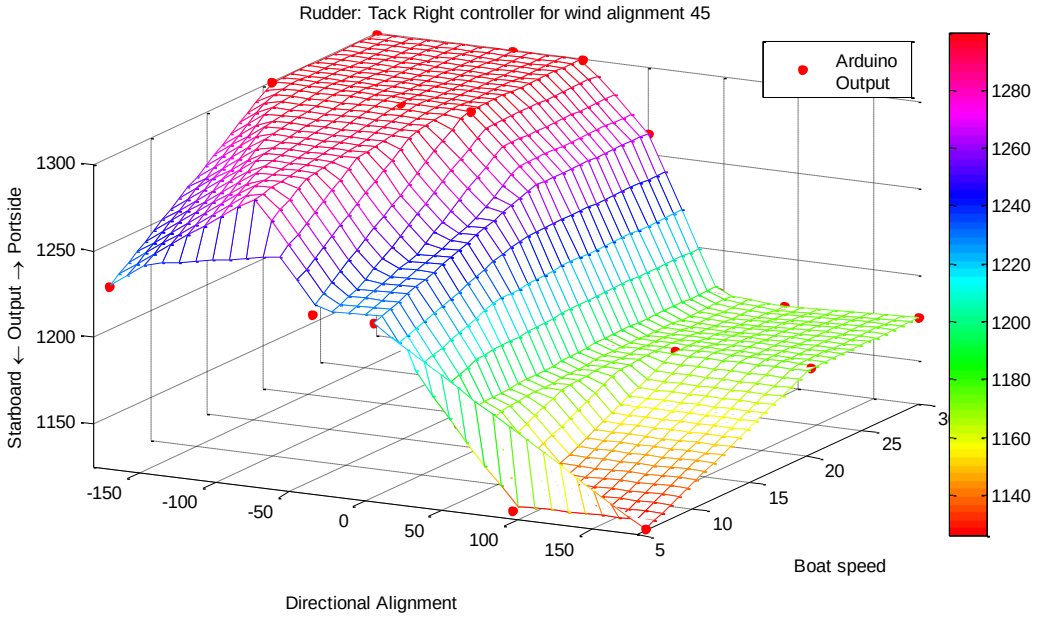


Figure 6.16: Experimental results for the tack right controller part 2.

This last controller's strategy is symetric to that of the tack left controller of section 6.2.3. If the objective is to the right side of the vessel and through or in the upwind zone then this is the controller to act. With wind alignment 0 all this controller does is steering the rudder to the right (starboard) as shown in figure 6.15. With wind alignment at 45 it is possible to navigate, as so if the objective is still to the left of the vessel it is in or through the upwind zone and the vessel steers centered, avoiding this zone. If the objective is to the right of the vessel then it continues to steer to the right until a more appropriate controller is chosen.

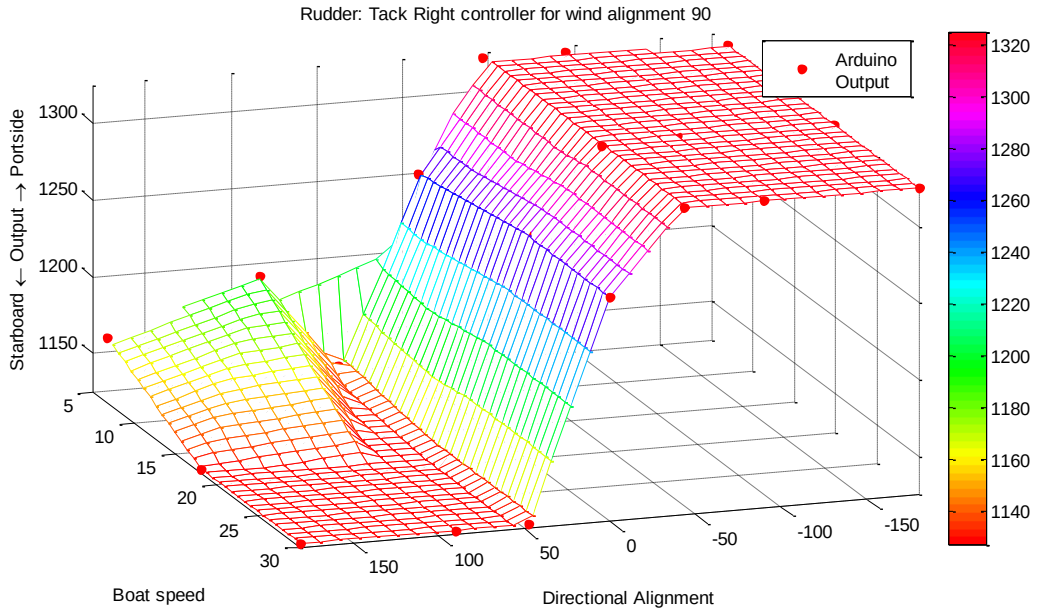


Figure 6.17: Experimental results for the tack right controller part 3.

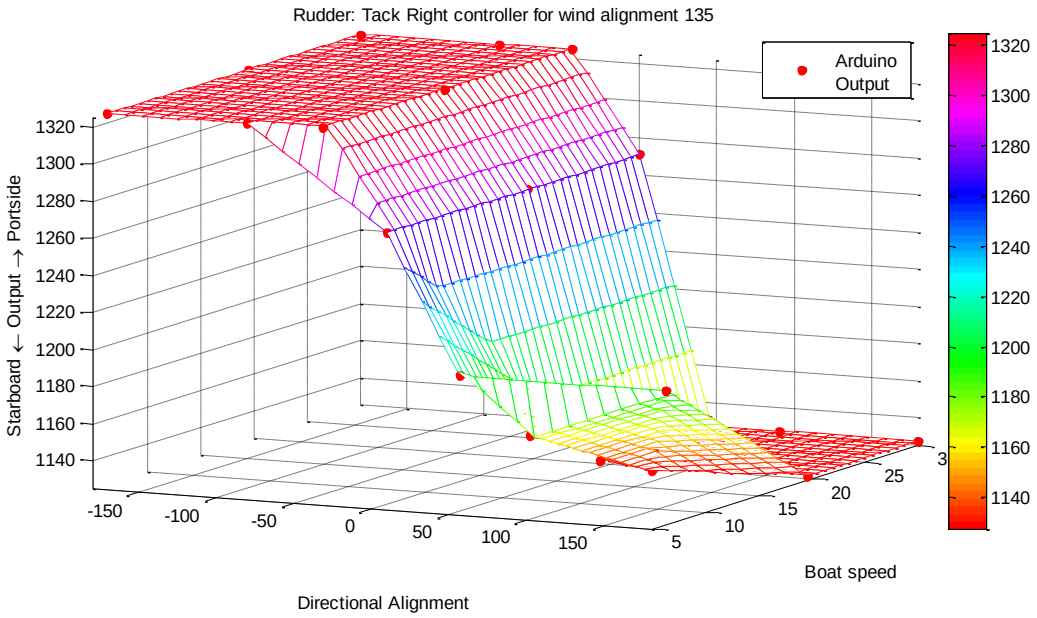


Figure 6.18: Experimental results for the tack right controller part 4.

Lateral wind conditions are similar to that of the wind alignment at 45 degrees. If the objective is to the left of the vessel then it continues centered until ready to make another tacking maneuver and if it is to the right the vessel simply steers into it. The specification to turn more smoothly when the speed is greater continues in all tack right results.

General result analysis

Analyzing the results it is safe to assume that all the controllers work as described in chapter 5, given the right inputs. Later tuning stages may be needed when sensors are available for field testing.

There is also the conclusion that, in all controllers both the outputs from the Arduino and Xfuzzy are practically identical, since the percentage error is very close to zero. The existing errors can be explained by the fact that both programs have different decimal precision.

7

Conclusions and future work

In this chapter, a summary of the work done and results achieved in this dissertation is given, as well as some directions for future research topics and ideas.

7.1 Conclusions

The main objective of the work presented in this thesis is to design the general layout of an autonomous sailboat controller using fuzzy logic, improvements suggested were also taken into account. To do this, different activities were performed and results were obtained, which are summarized next.

In section 5.1 a method to switch between manual and automatic control, using the vessel's original PWM signals is presented. Since the RF signals differ when the manual controller is turned on or off, this contrast was exploited. Implementation of a counter to use as the selector in a multiplexer was achieved. This way, it was possible to reduce the load on the Arduino's processor implementing most of the transition in hardware.

From sections 5.2 to 7.1 the general outline of the designed controller is presented. The objective was to design a fuzzy logic controller for an autonomous sailboat using an expert's approach. Following the knowledge obtained studying other cases [ASC97, Van97, SPJ07] and training in the Virtual Skipper Simulator the designed system was presented. First, most variables normally obtained using the available set of sensors where not intuitive to use and one of the purposes of fuzzy logic is intuitiveness, a new set of variables were designed. From these variables and following expert knowledge about sailing the four main controllers and their support structures were obtained. For the sail, since its only purpose is still speed gaining a single controller is enough. Since the rudder of a sailboat needs special navigation in windward zones more than one controller

is needed. To maintain simplicity, a controller was designed for each tacking maneuver, and another more general was designed. Rule bases for all controllers integrate smooth output response to avoid rough turns and consequent ship capsizing. The integration from Xfuzzy to Arduino code was successful, even though suffering minor problems explained in section 6.1.

As seen in the experimental results presented in chapter 6, all controllers behave as expected. Further softness of transitions from the output membership functions need to be tuned as soon as the Killer Whales Yatch is fully equipped with instrumentation sensors and available for testing.

7.2 Future Work

Following the work done in this dissertation and impossibilities due to lack of material, new investigation topics emerged. Some of these objectives are described below.

Higher level decider structure

Maritime races and courses do not need to be and usually are not linear. For instance, a race can have the same line for start and finish, making all the ships pass around a buoy and returning to their origin. In these cases, more than one course needs to be planned. On a later phase of this project, after successful testing of the controller on a vessel, a higher level planning structure can be designed. This structure's objective is to plan the general course of action, which can be a set of linear paths, detect when one path is finished and passing on to the next one until the last one is finished. Implementing a graphical interface application for planning and communication with the Arduino board is a possible course of action.

Other calculus options

Since all the calculi done in this work was done from scratch, space for improving is available. It can be suggested that when in tacking maneuvers, the vessel's maximum distance before tacking can be shortened while approaching the objective, in a manner shown in figure 7.1.

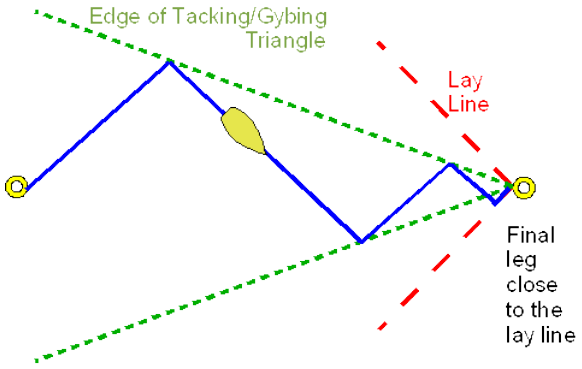


Figure 7.1: Alternative tacking approach [Saia].

This way, tacking maneuvers done too far when the objective is near are avoided.

Also, the distance to the objective is manually calculated as shown in section 5.4.4. NMEA protocol has a useful sentence to calculate this, the RMB (Recommended Minimum Navigation Information) sentence. It is useful for whenever a route or a goto is active, which can be the case.

Vessel - land communication

On a more advanced stage of optimization, before using the vessel on long term missions, the possibility to communicate with land while testing is available. For this, a small wireless router can be integrated with the system. On an early phase, research of available models for this objective was done, resulting in table 7.1.

Table 7.1: Available router model research.

Name	Size (cm)	Consumption	Supported protocols	Interfaces	Price range (Euros)
Apple Airport Express	94 x 7.5 x 2.85	46W(?)	802.11a/b/g/n	Ethernet gateway / printer USB connection / integrated charger / 3.5mm audio minijack,	97-124
D-Link DAP-1350	9.1 x 6.6 x 2	Specification not available	802.11n/g 802.3u	LAN/WAN gateway / Access Point / Wireless client / usb connection for Shareport	68-113
D-Link DWL-G730AP	8 x 6 x 1,7	6W	802.11b/g 802.3/u	Specification not available	53 (?)
Asus WL-330GE	8,6 x 6,2 x 1,7	5W Max (?)	802.11b/d/g 802.3/u 802.1X WPA WMM IPV4	Ethernet gateway / Access point / Wireless Client / Universal Repeater	31-82
Asus WL-330N3G	9 x 3,4 x 1,3	10W Max (?)	802.11n	Ethernet gateway / Access point / Wireless Client / Universal Repeater / Hotspot / 3G sharing	53-97
Linksys WTR54GS	10,7 x 7,3 x 3,1	Specification not available	802.11b/g 802.3/u	Ethernet gateway	100-212

Bibliography

[AC09] Jose C. Alves and Nuno A. Cruz. Um sistema computacional reconfigurável embarcado num veleiro autónomo. *V Jornadas sobre Sistemas Reconfiguráveis - REC'2009*, pages 28–35, 2009.

[Alo] Sanjay Krishnankutty Alonso. Structure of a fuzzy inference system. `http://www.dma.fi.upm.es/java/fuzzy/fuzzyinf/introfis_en.htm`. [Online; accessed 25-March-2013].

[ASC97] Jaime Abril, Jaime Salom, and Oscar Calvo. Fuzzy control of a sailboat. *Int. J. Approx. Reasoning*, 16(3-4):359–375, 1997.

[Ass] Historical Naval Ships Association. Hitting a moving target from a moving ship. `http://www.hnsa.org/doc/firecontrol/partd.htm`. [Online; accessed 25-March-2013].

[AT] Atmark-Techno. Suzaku-sz130-u00, hardware manual (english version) v1.0.2. `http://www.atmark-techno.com`. [Online; accessed 22-November-2008].

[ATA02] Martijn L. Van Aartrijk, Claudio P. Tagliola, and Pieter W. Adriaans. Ai on the ocean: the robosail project. In *In Proceedings of the 15th European Conference on Artificial Intelligence, ECAI*, pages 653–657, 2002.

[ATX01] C. Yang A. Tiano, A. Zirilli and C. Xiao. A neural autopilot for sailing yachts. *Proceedings of the 9th Mediterranean Conference on Control and Automation*, pages 27–29, 2001.

[Aus] Inc. Austriamicrosystems. As5040 10 bit 360^o programmable magnetic rotary encoder. `http://www.farnell.com/datasheets/77802.pdf`. [Online; accessed 15-July-2013].

[Ban] Maximo et Al Banzi. Arduino mega 2560. `http://arduino.cc/en/Main/ArduinoBoardMega2560`. [Online; accessed 23-May-2013].

[Ber92] HamidR. Berenji. Fuzzy logic controllers. In RonaldR. Yager and LotfiA. Zadeh, editors, *An Introduction to Fuzzy Logic Applications in Intelligent Systems*, volume 165 of *The Springer International Series in Engineering and Computer Science*, pages 69–96. Springer US, 1992.

[BHB] Hernando Barragan, Brett Hagman, and Alexander Brevig. Wiring. `http://wiring.org.co/`. [Online; accessed 23-May-2013].

[BJ12] Fabrice Le Bars and Luc Jaulin. An experimental validation of a robust controller with the vaimos autonomous sailboat. *Proceedings of the 5th International Robotic Sailing Conference*, 2012.

[Boa] BoaterEducation. Chapter 1: Know your boat: Sailboats. `http://www.boat-ed.com/nh/course/p1-6_knowyoursailboat.htm`. [Online; accessed 25-March-2013].

[Bri11] Y. Briere. Sailing robot performance: maximum speed tracking vs energy efficiency. *International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, pages 102–109, 2011.

[Cat] Tom Catalini. Sailing into the wind. `www.tomcatalini.com/sailing-into-the-wind/`. [Online; accessed 25-March-2013].

[Cor12] Atmel Corporation. 8-bit atmel microcontroller with 64k/128k/256k bytes in-system programmable flash. Data sheet, Atmel Corporation, 2012.

[Dez13] Elena Deza. *Encyclopedia of Distances*. 2013.

[dMdSa] Instituto de Microelectronica de Sevilla. The c++ code generation tool - xfcpp. `http://www2.imse-cnm.csic.es/Xfuzzy/Xfuzzy_3.3/tools/xfcpp.html`. [Online; accessed 26-May-2013].

[dMdSb] Instituto de Microelectronica de Sevilla. The inference monitor tool - xfmt. `http://www2.imse-cnm.csic.es/Xfuzzy/Xfuzzy_3.3/tools/xfmt.html`. [Online; accessed 26-May-2013].

[dMdSc] Instituto de Microelectronica de Sevilla. An overview of xfuzzy 3. `http://www2.imse-cnm.csic.es/Xfuzzy/Xfuzzy_3.3/index.html`. [Online; accessed 26-May-2013].

[dMdSd] Instituto de Microelectronica de Sevilla. The system edition tool - xfedit. `http://www2.imse-cnm.csic.es/Xfuzzy/Xfuzzy_3.3/tools/xfedit.html`. [Online; accessed 26-May-2013].

[EK06] G. Elkaim and R. Kelbley. Station keeping and segmented trajectory control of a wind-propelled autonomous catamaran. *45th IEEE Conference on Decision and Control*, 13(15):2424–2429, 2006.

[Ele] Robot Electronics. Cmps10 - tilt compensated compass module. http://www.robot-electronics.co.uk/htm/cmps10doc.htm. [Online; accessed 25-May-2013].

[Ent] Ubisoft Entertainment. Virtual skipper 5. http://www.virtualskipper.com/. [Online; accessed 25-March-2013].

[FR] Ben Fry and Casey et Al Reas. Processing. http://www.processing.org/. [Online; accessed 23-May-2013].

[Gil13] Flávio Gil. eventos - arquitectura de controlo distribuido para veleiro autónomo. Master's thesis, Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa, 2013. In progress.

[Glo] GlobalSat. Product user manual gps receiver engine board em-406a. https://www.sparkfun.com/datasheets/GPS/EM-406A_User_Manual.PDF. [Online; accessed 25-May-2013].

[Kha10] Dinesh Khattar. *The Pearson Guide to Complete Mathematics for AIEEE*. 2010.

[Kna] R. Benjamin Knapp. Fuzzy inference systems (mamdani). http://www.cs.princeton.edu/courses/archive/fall07/cos436/HIDDEN/Knapp/fuzzy004.htm. [Online; accessed 25-March-2013].

[KSJM09] H Klinck, K Stelzer, K Jafarmadar, and DK Mellinger. *Aas endurance: An autonomous acoustic sailboat for marine mammal research*. 2009.

[LB87] Hector J. Levesque and Ronald J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3:78–93, 1987.

[MA75] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Machine Studies*, 7 (1):1–13, 1975.

[Mica] Inc. Microchip. Pic24f family reference manual section 21. uart. http://ww1.microchip.com/downloads/en/DeviceDoc/en026583.pdf. [Online; accessed 15-July-2013].

[Micb] Inc. Microsoft. What is com? http://www.microsoft.com/com/default.mspx. [Online; accessed 15-July-2013].

[Micc] Microtransat. The microtransat challenge. http://www.microtransat.org/. [Online; accessed 28-March-2013].

[Moo82] Robert C. Moore. The role of logic in knowledge representation and commonsense reasoning. Technical Report 264, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025, Jun 1982.

[MVBBSS] F.J. Moreno-Velo, I. Baturone, A. Barriga, and S. Ságnchez-Solano.

[NAC08] José C. Alves Nuno A. Cruz. Ocean sampling and surveillance using autonomous sailboats. *Journal of the Osterreichische Gesellshaft fur Artificial Intelligence (Austrian Society for Artificial Inteligence)*, pages 25–31, 2008.

[Neg85] C.V. Negoita. *Expert systems and fuzzy systems*. The Benjamin/Cummings series in computer sciences. Benjamin/Cummings Pub. Co., 1985.

[NI] Inc. National Instruments. What is serial synchronous interface (ssi)? http://digital.ni.com/public.nsf/websearch/862567530005F09C862566BE004E469D?opendocument&Submitted&&node=133020_US. [Online; accessed 15-July-2013].

[OGfiC] INNOC Österreichische Gesellschaft für innovative Computerwissenschaften. World robotic sailing championship 2008. http://www.roboticsailing.org/en/2008. [Online; accessed 30-May-2013].

[PTRa] PTRobotics. 20 channel em-406a sirf iii gps receiver with antenna. http://www.ptrobotics.com/product.php?id_product=829. [Online; accessed 25-May-2013].

[PTRb] PTRobotics. Cmps10 - tilt compensated magnetic compass. http://www.ptrobotics.com/product.php?id_product=1195. [Online; accessed 25-May-2013].

[RRGI+11] M.A. Romero Ramirez, Y. Guo, S.H. Ieng, F. Plumet, R Benosman, and B. Gas. Omni-directional camera and fuzzy logic path planner for autonomous sailboat navigation. *Research in Computing Science, Special Issue in Advances in Computer Science and Electronic Systems*, 52:335–346, 2011.

[SA13] Ricardo Silva and Rita Alhandra. Controladores digitais e controlo difuso na navegação autónoma de veleiros - relatório final do piic. 2013. Internal Document.

[Saia] Ocean Sail. When to tack or gybe. http://www.oceansail.co.uk/Articles/VMGArticle.html.

[Saib] Spinnaker Sailing. The sail as an airfoil - sailing upwind. http://spinnaker-sailing.com/online-courses/lesson-1/sailing-upwind. [Online; accessed 25-March-2013].

[Sal96] Jim Saltonstall. *This is Sailing: A Complete Course*. Adlard Coles Nautical, fourth edition, 1996.

[SJ12] Roland Stelzer and Karim Jafarmadar. The robotic sailing boat asv roboat as a maritime research platform. In *Proceedings of 22nd International HISWA Symposium*, 2012.

[SP08] Roland Stelzer and Tobias Pröll. Autonomous sailboat navigation for short course racing. *Robot. Auton. Syst.*, 56(7):604–614, 2008.

[SPJ07] Roland Stelzer, Tobias Proll, and Robert Ivor John. Fuzzy logic control system for autonomous sailboats. *FUZZ-IEEE*, pages 1–6, 2007.

[ST] Inc. Sirf Technology. Nmea reference manual. `https://www.sparkfun.com/datasheets/GPS/NMEA%20Reference%20Manual1.pdf`. [Online; accessed 03-July-2013].

[Sta] National Marine Electronics Association 0183 Standard. Nmea 0183 standard. `http://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp`. [Online; accessed 03-July-2013].

[Sug85] Michio Sugeno. *Industrial Applications of Fuzzy Control*. Elsevier Science Inc., New York, NY, USA, 1985.

[Sysa] Argent Data Systems. Weather sensor assembly p/n 80422. `https://www.sparkfun.com/datasheets/Sensors/Weather/Weather%20Sensor%20Assembly..pdf`. [Online; accessed 25-May-2013].

[Sysb] Argent Data Systems. Wind / rain sensor assembly. `https://www.argentdata.com/catalog/product_info.php?products_id=145`. [Online; accessed 25-May-2013].

[Van97] T W Vaneck. Fuzzy guidance controller for an autonomous boat. *IEEE Control Systems*, pages 43–51, 1997.

[Vau] Bruce Vaughan. Polar-rectangular conversion. `http://www.teacherschoice.com.au/Maths_Library/Coordinates/polar_-_rectangular_conversion.htm`. [Online; accessed 30-May-2013].

[VBSB03] F.J.M. Velo, I. Baturone, S.S. Solano, and A. Barriga. Rapid design of fuzzy systems with xfuzzy. *The 12th IEEE International Conference on Fuzzy Systems*, 1:342 – 347, 2003.

[War91] W. H. Warden. A control system model for autonomous sailboat navigation. *IEEE Proc. of Southeastcon'91*, pages 944,947, 1991.

[Yal] Yaldex. Fuzzification. `http://www.yaldex.com/game-development/1592730043_ch33lev1sec2.html`. [Online; accessed 25-March-2013].

[YLZ95] J. Yen, R. Langari, and L.A. Zadeh. *Industrial applications of fuzzy logic and intelligent systems*. IEEE Press, 1995.

[Zad73] Lotfi A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-3(1):28 –44, jan. 1973.

[Zad83] L. A. Zadeh. The role of fuzzy logic in the management of uncertainty in expert systems. *Fuzzy Sets Syst.*, 11(1-3):197–198, January 1983.

[Zad86] L. A. Zadeh. Test-score semantics as a basis for a computational approach to the representation of meaning. *Literacy Linguistic Computing*, (1):24–35, 1986.

[Zad89] L.A. Zadeh. Knowledge representation in fuzzy logic. *IEEE Transactions on Knowledge and Data Engineering*, 1(1):89–100, 1989.

[Zim91] H.J. Zimmermann. *Fuzzy Set Theory and its applications*. Kluwer Academic, Dordrecht, 1991.

Experimental results tables

Table A.1: Sail controller experimental results.

Input			Output		
Rdir_align	Rboatspeed	Rwind_align	Xfuzzy	Arduino	Percentage error
15,3	9,9	0	17,07	17,07	0
15,3	19,5	0	1,512	1,51	0,15
15,3	30	0	1,512	1,51	0,15
30,6	9,9	0	17,07	17,07	0
30,6	20,1	0	1,512	1,51	0,15
30,6	30	0	1,512	1,51	0,15
45	9,6	0	56,51	56,51	0
45	19,5	0	48,6	48,6	0
45	30	0	48,6	48,6	0
60,3	10,2	0	88,2	88,2	0
60,3	19,2	0	74,5	74,5	0
60,3	30	0	74,5	74,5	0
74,7	9	0	94,12	94,12	0
74,7	19,5	0	74,5	74,5	0
74,7	30	0	74,5	74,5	0
90	10,2	0	88,2	88,2	0
90	19,8	0	74,5	74,5	0
90	30	0	74,5	74,5	0

Input			Output		
Rdir_align	Rboatspeed	Rwind_align	Xfuzzy	Arduino	Percentage error
0	9,6	45	34,17	34,02	0,44
0	19,8	45	35,98	35,98	0
0	30	45	35,98	35,98	0
15,3	9,6	45	34,02	34,02	0
15,3	19,5	45	35,98	35,98	0
15,3	30	45	35,98	35,98	0
30,6	9,9	45	32,47	32,48	0
30,6	19,5	45	35,98	35,98	0
30,6	30	45	35,98	35,98	0
45	10,5	45	51,43	51,43	0
45	19,5	45	49,12	49,12	0
45	30	45	49,12	49,12	0
60,3	9,6	45	91,36	91,36	0
60,3	20,1	45	74,5	74,5	0
60,3	30	45	74,5	74,5	0
74,7	10,2	45	88,2	88,2	0
74,7	21,9	45	74,5	74,5	0
74,7	30	45	74,5	74,5	0
90	10,2	45	88,2	88,2	0
90	19,8	45	74,5	74,5	0
90	30	45	74,5	74,5	0
0	9,9	90	50	50	0
0	20,1	90	50	50	0
0	30	90	50	50	0
15,3	5,1	90	49,9	50	0
15,3	9,9	90	50	50	0
15,3	22,5	90	49,9	50	0
30,6	9,6	90	50	50	0
30,6	18,3	90	49,999	50	0
30,6	30	90	49,999	50	0
45	12	90	60,8	60,8	0
45	18,9	90	63,8	63,8	0
45	30	90	63,8	63,8	0
60,3	10,2	90	88,2	88,2	0
60,3	20,1	90	74,5	74,5	0
60,3	30	90	74,5	74,5	0
74,7	9,6	90	91,36	91,36	0
74,7	19,8	90	74,5	74,5	0
74,7	30	90	74,5	74,5	0
90	9,6	90	91,36	91,36	0
90	19,5	90	74,5	74,5	0
90	30	90	74,5	74,5	0
0	9,6	135	50	50	0
0	19,8	135	49,9	50	0
0	30	135	49,9	50	0
15,3	30	135	49,9	50	0
15,3	20,7	135	49,9	50	0
15,3	10,2	135	50	50	0

Input			Output		
Rdir_align	Rboatspeed	Rwind_align	Xfuzzy	Arduino	Percentage error
30,6	10,1	135	50	50	0
30,6	20,7	135	49,9	50	0
30,6	30	135	49,9	50	0
45	9,9	135	71,9	71,9	0
45	19,8	135	63,8	63,8	0
45	30	135	63,8	63,8	0
60,3	9,9	135	89,8	89,8	0
60,3	19,5	135	74,5	74,5	0
60,3	30	135	74,5	74,5	0
74,7	9,6	135	91,3	91,3	0
74,7	20,4	135	74,5	74,5	0
74,7	30	135	74,5	74,5	0
90	9,9	135	89,8	89,8	0
90	19,8	135	74,5	74,5	0
90	30	135	74,5	74,5	0
0	9,9	180	89,8	89,8	0
0	20,4	180	74,5	74,5	0
0	30	180	74,5	74,5	0
15,3	9,6	180	91,4	91,4	0
15,3	19,5	180	74,5	74,5	0
15,3	30	180	74,5	74,5	0
30,6	9,6	180	91,4	91,4	0
30,6	20,4	180	74,5	74,5	0
30,6	30	180	74,5	74,5	0
45	10,5	180	86,47	86,47	0
45	20,1	180	74,5	74,5	0
45	30	180	74,5	74,5	0
60,3	10,2	180	88,2	88,2	0
60,3	19,8	180	74,5	74,5	0
60,3	30	180	74,5	74,5	0
74,7	10,5	180	86,47	86,47	0
74,7	19,2	180	74,5	74,5	0
74,7	30	180	74,5	74,5	0
90	9,9	180	89,83	89,83	0
90	20,1	180	74,5	74,5	0
90	30	180	74,5	74,5	0

Table A.2: Rudder: Favorable Wind controller experimental results.

Input			Output		
Rdir_align	Rboatspeed	Rwind_align	Xfuzzy	Arduino	Percentage error
-180	6	0	1127,4	1127,45	0
-180	19,8	0	1226,8	1226,87	0
-180	30	0	1226,8	1226,87	0
-90	5,7	0	1127,4	1127,45	0
-90	20,4	0	1226,8	1226,87	0
-90	30	0	1226,8	1226,87	0
-43,199	6	0	1127,4	1127,45	0
-43,199	19,5	0	1226,8	1226,87	0
-43,199	30	0	1226,8	1226,87	0
0	6	0	1338,9	1340	0,1
0	19,8	0	1339	1339	0
0	30	0	1339	1339	0
46,8	5,7	0	1553,1	1554,15	0,1
46,8	20,4	0	1451,1	1451,19	0
46,8	30	0	1451,1	1451,19	0
90	6	0	1553,1	1553,15	0
90	19,8	0	1451,1	1451,19	0
90	30	0	1451,1	1451,19	0
180	6	0	1553,1	1553,15	0
180	19,8	0	1451,1	1451,19	0
180	30	0	1451,1	1451,19	0
-180	5,7	45	1553,1	1553,15	0
-180	21	45	1451,1	1451,19	0
-180	30	45	1451,1	1451,19	0
-90	6	45	1553,1	1553,15	0
-90	18,3	45	1451,1	1451,19	0
-90	30	45	1451,1	1451,19	0
-43,199	6	45	1451,1	1451,19	0
-43,199	18,3	45	1451,1	1451,19	0
-43,199	30	45	1451,1	1451,19	0
0	5,7	45	1325	1325	0
0	21	45	1325	1325	0
0	30	45	1325	1325	0
46,8	6	45	1226,8	1226,87	0
46,8	19,2	45	1226,8	1226,87	0
46,8	30	45	1226,8	1226,87	0
90	4,5	45	1127,4	1127,45	0
90	18,3	45	1226,8	1226,87	0
90	30	45	1226,8	1226,87	0
180	4,8	45	1127,4	1127,45	0
180	21,9	45	1226,8	1226,87	0
180	30	45	1226,8	1226,87	0
-180	5,1	90	1553,1	1553,15	0
-180	19,2	90	1451,1	1451,19	0
-180	30	90	1451,1	1451,19	0

Input			Output		
Rdir_align	Rboatspeed	Rwind_align	Xfuzzy	Arduino	Percentage error
-90	4,8	90	1553,1	1553,15	0
-90	18,3	90	1451,1	1451,19	0
-90	30	90	1451,1	1451,19	0
-43,199	4,8	90	1451,1	1451,19	0
-43,199	21,9	90	1451,1	1451,19	0
-43,199	30	90	1451,1	1451,19	0
0	5,1	90	1324,9	1325,01	0
0	19,2	90	1325	1325	0
0	30	90	1325	1325	0
46,8	5,7	90	1226,8	1226,87	0
46,8	18,3	90	1226,8	1226,87	0
46,8	30	90	1226,8	1226,87	0
90	6	90	1127,4	1127,45	0
90	21,9	90	1226,8	1226,87	0
90	30	90	1226,8	1226,87	0
180	4,8	90	1127,4	1127,45	0
180	19,2	90	1226,8	1226,87	0
180	30	90	1226,8	1226,87	0
-180	4,8	135	1553,1	1553,15	0
-180	19,2	135	1451,1	1451,19	0
-180	30	135	1451,1	1451,19	0
-90	6	135	1553,1	1553,15	0
-90	21	135	1451,1	1451,19	0
-90	30	135	1451,1	1451,19	0
-43,199	4,5	135	1451,1	1451,19	0
-43,199	19,2	135	1451,1	1451,19	0
-43,199	30	135	1451,1	1451,19	0
0	5,1	135	1324,9	1325,01	0
0	17,4	135	1325	1325	0
0	30	135	1325	1325	0
46,8	6	135	1226,8	1226,87	0
46,8	21	135	1226,8	1226,87	0
46,8	30	135	1226,8	1226,87	0
90	5,7	135	1127,4	1127,45	0
90	19,2	135	1226,8	1226,87	0
90	30	135	1226,8	1226,87	0
180	5,1	135	1127,4	1127,45	0
180	17,4	135	1226,8	1226,87	0
180	30	135	1226,8	1226,87	0
-180	6	180	1647,4	1647,46	0
-180	21	180	1553,1	1553,15	0
-180	30	180	1553,1	1553,15	0
-90	5,7	180	1647,4	1647,46	0
-90	18,9	180	1553,1	1553,15	0
-90	30	180	1553,1	1553,15	0
-43,199	5,1	180	1553,1	1553,15	0
-43,199	17,4	180	1451,1	1451,19	0
-43,199	30	180	1451,2	1451,19	0

Input			Output		
Rdir_align	Rboatspeed	Rwind_align	Xfuzzy	Arduino	Percentage error
0	6,9	180	1324,9	1325,01	0
0	21	180	1325	1325	0
0	30	180	1325	1325	0
46,8	6	180	1127,4	1127,45	0
46,8	18,3	180	1226,8	1226,87	0
46,8	30	180	1226,8	1226,87	0
90	5,1	180	1025,4	1025,49	0
90	19,5	180	1127,4	1127,45	0
90	30	180	1127,4	1127,45	0
180	6	180	1025,4	1025,49	0
180	21	180	1127,4	1127,45	0
180	30	180	1127,4	1127,45	0

Table A.3: Rudder: Tack Right controller experimental results.

Input			Output		
Rdir_align	Rboatspeed	Rwind_align	Xfuzzy	Arduino	Percentage error
-180	9,3	0	1226,8	1226,8	0
-180	20,4	0	1226,8	1226,8	0
-180	30	0	1226,8	1226,8	0
-90	10,2	0	1226,8	1226,8	0
-90	19,8	0	1226,8	1226,8	0
-90	30	0	1226,8	1226,8	0
-43,199	10,2	0	1226,8	1226,8	0
-43,199	19,8	0	1226,8	1226,8	0
-43,199	30	0	1226,8	1226,8	0
0	6	0	1025,4	1025,4	0
0	19,8	0	1226,8	1226,8	0
0	30	0	1226,8	1226,8	0
46,8	5,7	0	1025,4	1025,4	0
46,8	20,4	0	1226,8	1226,8	0
46,8	30	0	1226,8	1226,8	0
90	5,7	0	1025,4	1025,4	0
90	20,1	0	1226,8	1226,8	0
90	30	0	1226,8	1226,8	0
180	5,7	0	1025,4	1025,4	0
180	20,7	0	1226,8	1226,8	0
180	30	0	1226,8	1226,8	0

Input			Output		
Rdir_align	Rboatspeed	Rwind_align	Xfuzzy	Arduino	Percentage error
-180	6,3	45	1225,3	1225,3	0
-180	20,7	45	1299,8	1299,8	0
-180	30	45	1299,8	1299,8	0
-90	6,9	45	1282,7	1282,7	0
-90	20,1	45	1299,8	1299,8	0
-90	30	45	1299,8	1299,8	0
-43,199	6	45	1225,1	1225,1	0
-43,199	20,1	45	1299,8	1299,8	0
-43,199	30	45	1299,8	1299,8	0
0	5,7	45	1225,5	1225,5	0
0	19,8	45	1261,1	1262,1	0
0	30	45	1261,9	1262	0
46,8	6,9	45	1169,1	1169,1	0
46,8	20,4	45	1174,8	1174,9	0
46,8	30	45	1174,8	1174,9	0
90	6	45	1126,1	1126,1	0
90	20,4	45	1174,8	1174,9	0
90	30	45	1174,8	1171,9	0,24
180	5,7	45	1126,1	1126,1	0
180	20,4	45	1174,8	1174,9	0
180	30	45	1174,8	1174,9	0
-180	6,9	90	1324,9	1325	0
-180	19,8	90	1325	1325	0
-180	30	90	1325	1325	0
-90	6	90	1324,9	1325	0
-90	19,5	90	1325	1325	0
-90	30	90	1325	1325	0
-43,199	5,7	90	1324,9	1325	0
-43,199	20,1	90	1325	1325	0
-43,199	30	90	1325	1325	0
0	6,9	90	1257,9	1258	0
0	19,8	90	1270,7	1270,9	0
0	30	90	1270,7	1270,9	0
46,8	6,9	90	1137,3	1137,3	0
46,8	19,5	90	1127,4	1127,4	0
46,8	30	90	1127,4	1127,4	0
90	6,3	90	1197	1197	0
90	19,5	90	1127,4	1127,4	0
90	30	90	1127,4	1127,4	0
180	6,6	90	1167,2	1167,2	0
180	18	90	1127,4	1127,4	0
180	30	90	1127,4	1127,4	0
-180	6,3	135	1324,9	1325	0
-180	19,5	135	1325	1325	0
-180	30	135	1325	1325	0
-90	6,3	135	1324,9	1325	0
-90	19,5	135	1325	1325	0
-90	30	135	1325	1325	0

Input			Output		
Rdir_align	Rboatspeed	Rwind_align	Xfuzzy	Arduino	Percentage error
-43,199	6,6	135	1324,9	1325	0
-43,199	18	135	1325	1325	0
-43,199	30	135	1325	1325	0
0	6,3	135	1271,6	1271,6	0
0	19,5	135	1270,7	1270,9	0
0	30	135	1270,7	1270,9	0
46,8	6,3	135	1197	1197	0
46,8	19,5	135	1127,4	1127,4	0
46,8	30	135	1127,4	1127,4	0
90	6,6	135	1167,2	1167,2	0
90	18	135	1127,4	1127,4	0
90	30	135	1127,4	1127,4	0
180	6,3	135	1197	1197	0
180	19,5	135	1127,4	1127,4	0
180	30	135	1127,4	1127,4	0

Table A.4: Rudder: Tack Left controller experimental results.

Input			Output		
Rdir_align	Rboatspeed	Rwind_align	Xfuzzy	Arduino	Percentage error
-180	6,6	0	1590,8	1590,8	0
-180	20,4	0	1553,1	1553,1	0
-180	30	0	1553,1	1553,1	0
-90	5,7	0	1647,4	1647,4	0
-90	19,8	0	1553,1	1553,1	0
-90	30	0	1553,1	1553,1	0
-43,199	6,3	0	1619,2	1619,17	0
-43,199	19,8	0	1553,1	1553,1	0
-43,199	30	0	1553,1	1553,1	0
0	6,6	0	1590,8	1590,8	0
0	19,8	0	1553,1	1553,1	0
0	30	0	1553,1	1553,1	0
46,8	5,7	0	1647,4	1647,4	0
46,8	20,4	0	1553,1	1553,1	0
46,8	30	0	1553,1	1553,1	0
90	6	0	1647,4	1647,4	0
90	19,8	0	1553,1	1553,1	0
90	30	0	1553,1	1553,1	0
180	5,7	0	1647,4	1647,4	0
180	19,5	0	1553,1	1553,1	0
180	30	0	1553,1	1553,1	0

Input			Output		
Rdir_align	Rboatspeed	Rwind_align	Xfuzzy	Arduino	Percentage error
-180	6	45	1600,2	1600,3	0
-180	20,4	45	1485,1	1485,1	0
-180	30	45	1485,1	1485,1	0
-90	6,6	45	1538,4	1538,4	0
-90	19,8	45	1485,1	1485,1	0
-90	30	45	1485,1	1485,1	0
-43,199	5,7	45	1600,2	1600,3	0
-43,199	19,5	45	1485,1	1485,1	0
-43,199	30	45	1485,1	1485,1	0
0	6	45	1462,6	1462,6	0
0	20,4	45	1406,4	1406,4	0
0	30	45	1406,4	1406,4	0
46,8	5,7	45	1432,4	1432,4	0
46,8	19,8	45	1383,4	1383,4	0
46,8	30	45	1383,4	1383,4	0
90	5,7	45	1432,4	1432,4	0
90	19,5	45	1383,4	1383,4	0
90	30	45	1383,4	1383,4	0
180	6	45	1432,4	1432,4	0
180	20,4	45	1383,4	1383,4	0
180	30	45	1383,4	1383,4	0
-180	5,7	90	1553,1	1553,1	0
-180	19,8	90	1451,1	1451,1	0
-180	30	90	1451,1	1451,1	0
-90	7,5	90	1553,1	1553,1	0
-90	19,5	90	1451,1	1451,1	0
-90	30	90	1451,1	1451,1	0
-43,199	6	90	1553,1	1553,1	0
-43,199	20,4	90	1451,1	1451,1	0
-43,199	30	90	1451,1	1451,1	0
0	5,7	90	1401	1401,05	0
0	19,5	90	1376,4	1376,4	0
0	30	90	1376,4	1376,4	0
46,8	5,7	90	1324,9	1325	0
46,8	19,5	90	1325	1325	0
46,8	30	90	1325	1325	0
90	6	90	1324,9	1325	0
90	20,4	90	1325	1325	0
90	30	90	1323	1323	0
180	5,7	90	1324,9	1325	0
180	19,5	90	1325	1325	0
180	30	90	1325	1325	0
-180	6	135	1553,1	1553,1	0
-180	19,5	135	1451,1	1451,1	0
-180	30	135	1451,1	1451,1	0
-90	6	135	1553,1	1553,1	0
-90	20,4	135	1451,1	1451,1	0
-90	30	135	1451,1	1451,1	0

Input			Output		
Rdir_align	Rboatspeed	Rwind_align	Xfuzzy	Arduino	Percentage error
-43,199	6,6	135	1491,9	1491,9	0
-43,199	19,8	135	1451,1	1451,1	0
-43,199	30	135	1451,1	1451,1	0
0	5,7	135	1401	1401	0
0	19,5	135	1376,4	1376,4	0
0	30	135	1376,4	1376,4	0
46,8	6	135	1324,9	1325	0
46,8	20,4	135	1325	1325	0
46,8	30	135	1325	1325	0
90	6	135	1324,9	1325	0
90	19,8	135	1325	1325	0
90	30	135	1325	1325	0
180	5,7	135	1324,9	1325	0
180	19,5	135	1325	1325	0
180	30	135	1325	1325	0